# Department of Computer Science


# Technical Report

Undergraduate Dissertation: Dynamic Planning:: The Application of AI Planning Advances to Project Management

Meri Williams

**Technical Report 2005-07** **July 2005**

**ISSN 1740-9497**

**Contact Address:**
Department of Computer Science
University of Bath
Bath, BA2 7AY
United Kingdom
URL: http://www.cs.bath.ac.uk

**ISSN 1740-9497**

# Dynamic Planning :: The Application of AI Planning Advances to Project Management

Meri Williams (cs1mjaw)

BSc (Hons) Computer Information Systems

19th May 2005

# Dynamic Planning :: The Application of AI Planning Advances to Project Management

Submitted by Meri Williams

## COPYRIGHT



## DECLARATION

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed: .....

Dated: .....

This thesis may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation

Signed: .....

Dated: .....

# Abstract

Traditional project planning techniques and tools do not afford the flexibility needed to be effective in our modern, dynamic environments. Although in the field of artificial intelligence, planning has developed to take into account uncertainty and the need to adapt to change, there is little evidence of this in project management. In this dissertation we examine the fields of project management and artificial intelligence planning, applying advances in the latter to produce a new project planning methodology: Dynamic Planning. We present planning software specifically developed to support this new way of planning. Initial results of evaluating the new approach are also presented, which indicate that the new methodology is more flexible and better for tracking progress than traditional project planning techniques. Recommendations for further research and development are also discussed.

# Acknowledgements

Firstly, many thanks to my supervisor, Dr Joanna Bryson, for the original inspired concept of combining artificial intelligence and project planning, as well as gracefully allowing me to pick up her idea and run off with it in a different direction. Her support, feedback and RSI management tips were all invaluable.

Thanks also to my personal tutor, Dr Marina de Vos, and to Dr Alwyn Barry, for their understanding and advice, which made coping with various unfortunate events this year infinitely easier.

Thanks also to my army of proofreaders: Elly, Husey, Steve, Laurence and Simon. Without you this dissertation might have ended up eleggible.

Thank you to my parents for providing me with an excellent school education and my grandparents for making it possible for me to attend university in the UK. Yes, it appears I may well be getting a degree in that "computer thing I do".

On a slightly more serious and technical note, thank you to all those on the wxPython-users mailing list for their help and suggestions and particularly to Robin Dunn for creating the original product and all his hard work in maintaining it. Thanks also to the lyx-users mailing list for their help in tackling various document display issues.

This dissertation would not have been written without the music of Melissa Etheridge, Garbage, Neil Young & Crazy Horse, Johnny Clegg & Savuka and Sheryl Crow, the late night conversational ability of Molly Holzschlag or the hospitality of the staff at The Crown at Bathford. Thanks to all for loaned sanity.

Last, but by no means least, thanks to my fiancee, Elly, for her support, love, ability to spot a grammar mistake at 10 paces and for not minding my turning nocturnal for extended periods during the last 8 months.

# Some Thoughts on Plans

"He who every morning plans the transaction of the day and follows out that plan, carries a thread that will guide him through the maze of the most busy life. But where no plan is laid, where the disposal of time is surrendered merely to the chance of incidence, chaos will soon reign." *Victor Hugo (1802 - 1885)*

"In preparing for battle I have always found that plans are useless, but planning is indispensable." *Dwight D. Eisenhower (1890 - 1969)*

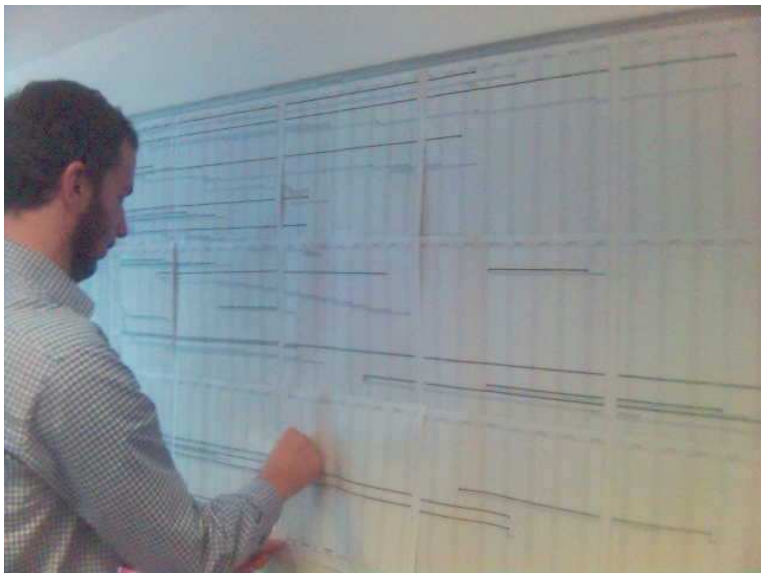

Figure 1: An Extreme Gantt Chart[1]

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Project planning is an integral part of project management, essential to ensure that the project delivers its desired results on time, within budget and to specification. The traditional approach to project planning intends to specify exactly what needs to be done at any given time in the project's lifecycle. This thorough approach seems logical and appropriate, which is why it is surprising that only a small number of projects are actually deemed to be successful [The Standish Group, 2005].

The key issue is that traditional project planning does not take account of the dynamic environment within which the project exists. Unforeseen occurrences add to the work that must be carried out, or present opportunities to achieve results faster or more cheaply. The dilemma for the project manager is evident: if they spend too much time planning and replanning, they may fail to manage the project properly. However, if they fail to adjust the plan to changing circumstances, it quickly becomes out-dated and useless.

A similar problem exists in the field of artificial intelligence (AI) planning. In order to create intelligent systems that can act independently, they must be instilled with the ability to plan; a mechanism for choosing an appropriate set of actions to achieve the eventual goal. Again, however, if too much time is spent analysing the situation and planning, then by the time an appropriate plan has been derived, the environment may have changed again, invalidating the plan.

Since both areas experience similar problems, one might expect the research in each area to be heading in similar directions. In fact, the opposite is true. Most of the recent work in project planning has focused on creative ways to force project managers and teams to follow the traditional process. Very few researchers seem to question the validity of this approach at all. In AI planning the picture is quite different. Here there has been a progression from classical planning (where environments are assumed to be steady, predictable and changeable only by direct intervention) through to approaches better geared for addressing "real world" problems (such as nondeterministic environments).

Reactive Planning is a prime example of the latter approach, with a focus on helping intelligent systems to choose the most appropriate next action. The short-term focus allows reactive systems to react quickly to changing conditions, at the same time still heading towards the eventual goal.

The original aim of this project was to develop planning software to afford project managers the same sort of flexibility as these reactive systems. However, during the background reading for the project, it soon became apparent that there was no existing project planning methodology that afforded this level of flexibility. The focus thus shifted to applying the recent advances in AI planning to project management, developing a new methodology for flexible project management.

The methodology that we present is Dynamic Planning, which draws on various planning approaches from both project management and artificial intelligence disciplines, as well as direct research into the way that people actually use plans. In order to evaluate the effectiveness of this new planning paradigm, the Dynamic Planner software was developed. Our initial results indicate that Dynamic Planning supports more flexible project planning, as well as more effective progress tracking.

This document is structured as follows: the next chapter covers the background reading and research into project and AI planning, including a discussion of the similarities and differences of the two fields. In Chapter 3 we propose the new Dynamic Planning methodology. Chapter 4 is concerned with the development of the Dynamic Planner and in Chapter 5 we present the results of our initial evaluation of the methodology, using this software. Finally, in Chapter 6 conclusions and further work are presented.

## 1.1 Project Aims & Objectives

The aim of this project is to *develop a method for flexible project planning and then develop software to support it, so that an initial evaluation of the methodology can be carried out.*

The objectives are as follows:

- Research and critique existing project planning methodologies
- Research and critique artificial intelligence planning techniques
- Develop a method for flexible project planning
- Develop software to support this method
- Evaluate the effectiveness of the flexible project planning methodology, using the software
- Gain technical, academic and managements skills:
  - Learn new programming language and gain experience in application development
  - Gain experience in academic research and writing
  - Develop an understanding of AI planning techniques
  - Gain an in-depth understanding of project planning and management techniques

# Chapter 2

# Research

## 2.1 Introduction

Before considering any new methodology that we might develop for flexible project planning, it is of course necessary to consider the current state of the art. In this chapter we will firstly examine project planning and then the various plan representations that are associated with existing approaches. We then go on to discuss AI planning, including traditional classical planning and more recent developments that aim to adjust to real world conditions.

First, however, we should look at the purpose of plans. Agre and Chapman [1989] suggest the way in which plans are designed to be used is very important. There are two main models: Plan-As-Program and Plan-As-Communication. Thinking of a plan as a program, the user of the plan is expected to execute it exactly – what they do will be precisely defined by the plan. When a plan is used for communication, then the action taken is not prescribed by the plan. Instead the plan is used as one factor in deciding what to do next.

There is a world of difference between the two. When a plan is used as a set of instructions to be executed, then far more detail is required than in a plan intended for communication. This is because every eventuality needs to be predicted and planned around, because all actions depend on the plan. Plans-As-Program also need to be extremely precise, again because the user is presumed to be unable to deal with uncertainty or diversion from the plan.

Plans-As-Communication can afford to be much less detailed and precise. Plans intended for communicating need only guide actions, since the plan itself will only be used as one input to the decision of what action to take. This absolves the planner of the need to predict the future – the identification of every eventuality required by the Plan-As-Program model. However, Plans-As-Program may also succumb to fragility and too much detail, unless the planner accepts that *perfect communication is impossible*[Cockburn, 2002a]. Perfect communication requires everything about a situation to be explicitly detailed. This is simply not practical. Instead, the goal of Plans-As-Communication should be to manage the incompleteness of communication well – that is, to communicate just well enough for the intended audience.

Looking at these two modes of plan use, we have inadvertently discussed another key element of plans – their level of precision[Cockburn, 2002a]. The extremely detailed execution plan of the Plan-As-Program mode represents a very high-precision plan. Plans-As-Communication in general can be lower precision, since they assume some knowledge and some ability to improvise on the part of the user. Cockburn [2002a] suggests that plans with different levels of precision are useful at different stages in a project, whereas the Systems-Gap-Working-Party [1984] suggests that different levels of detail are appropriate at different levels in the organisation.

During the course of our analysis of existing project planning methodologies, it will be useful to discuss each in terms of the plan model (Program vs Communication) and also the level of precision (low, medium or high). Both of these factors have an effect on the level of detail typically present in the plan, which we will also see when examining some project planning artifacts.

## 2.2 Project Planning in the Business World

### 2.2.1 Traditional Project Planning

Planning as described in the traditional POMA (Planning, Organising, Monitoring and Adjusting) management cycle [Tsui, 2004] is a very thorough affair that results in detailed plans from the outset. The key thread running through all of POMA is that the project must be controlled throughout its life cycle. The focus during each section highlights this: Planning defines every piece of work that must be done as well as when and by whom; Organising sets the team up to carry out the plan; Monitoring ensures that deviations from the plan are identified; and Adjusting rectifies these variations through relevant action.

POMA is described as just one methodology for project management, but upon inspection most other methodologies appear to be slightly varied articulations of the same approach. Although only Tsui [2004] specifically identifies POMA as the framework for the planning process then described, the same key facets are evident in standard management and software engineering texts, such as Lock [2001], Sommerville [2001], Brown [1998] and Kliem and Ludin [1993]. Other sources employ the same approach but with emphasis on different aspects – for instance, Goodpasture [2004] focuses on quantitative methods for estimation, monitoring, etc, whilst Kliem and Ludin [1992] are most concerned with the "people factor" inherent in any project where the actual results are achieved by a team.

Looking at the various texts available we can derive the following as the steps that make up the traditional project planning process :

- Defining project goals and objectives.
- Breaking down the work into its component tasks .
- Defining the dependencies between tasks.
- Estimating the effort to complete each task.

- Identifying the resources available.
- Constructing a schedule based on these effort & resource estimates.
- Identifying any resource clashes and "levelling" the plan to distribute the work evenly.

In theory, the plans produced by this process are excellent – they are highly detailed, clearly indicate the precedence of tasks, the schedule (which includes the projected delivery date, of course) as well as which resources are assigned to which tasks and who holds responsibility for deliverables. According to the Plan-As-Program paradigm, it should be possible for the project team to follow the plan to the letter from beginning to end and to deliver the project on time, to specification and within budget. Why then do some reports claim only 9 - 16% of projects actually achieve this feat? [The Standish Group, 2005]

Tsui [2004] appears to believe that the planning process is just not being followed well enough:

> "Project planning includes a time-consuming and very important set of tasks that is, unfortunately, often rushed. It is much wiser to spend the appropriate time needed to develop a good plan initially than to have to make multiple and costly adjustments later"

But he himself goes on to acknowledge:

> "Even with a well-conceived plan, it is unusual not to encounter some conditions that require unexpected changes during the project. However, having a well thought out plan facilitates making project adjustments even at a much later phase." [Tsui, 2004]

This appears to be a recurring theme. Even the staunchest advocates of the traditional planning approach admit that it is unheard of for plans to remain both accurate and unchanged throughout the course of the project. The primary reason for this is obvious: human beings cannot see into the future. At the start of a project it is very difficult to fully understand even the project itself, let alone predict changes in resources, requirements, the environment or any number of other factors. Brown [1998] summarises the issue:

> "Planning requires a large amount of information, and the amount and quality of information which you will have is inversely proportional to the length of time between when you plan and when the tasks should be executed"

So we can see that it is virtually impossible to create a complete, accurate Plan-As-Program at the beginning of the project. This leads us to the realisation that the plans will have to be updated as circumstances change, opportunities or setbacks

are discovered or original estimates prove to be inaccurate. Despite Tsui's assurance that a well thought out plan will make adjustments at a later stage easier, a number of sources warn against project managers spending all their time updating the schedule and not actually managing the project (Block and Frame [1998], Brown [1998], Anderson et al. [1988], DeMarco [1997])

Surely though the only certainty in a project is that things will change? If changes (and the updates and adjustments they necessitate) make plans too time-consuming to keep updated, should we not just give up on the idea of planning altogether?

Luckily, this is not the only option. If we look more closely at the output of traditional project planning, we realise that these plans are extremely high precision. So much detail is included that day-to-day tasks are planned months and years in advance. It is for this reason that updating the plans is so time-consuming, as alluded to by Lock [2001]:

> "Although it is possible to schedule more than 100 jobs on an adjustable bar chart, rescheduling is a different story. Setting a complex plan up in the first place might take a few working days or a week. Adjusting it subsequently to keep in step with changes might prove impossible"

In fact, the level of detail that is assumed to be needed, along with the time-consuming nature of subsequent updates, causes many project managers to avoid planning completely:

> "The more precision in the plan, the more fragile it is, which is why constructing Gantt charts is so feared: It is time-consuming to produce and gets out of date with the slightest surprise event" [Cockburn, 2002a]

So it would appear that there are many problems with traditional project planning: the initial planning process is very time-consuming and although it produces seemingly useful, detailed plans, these soon become out-of-date. Monitoring this, adjusting the plans or even re-planning completely (as is sometimes needed) is such a time-consuming process that it often becomes the project manager's primary task, eclipsing other activities. Worse still, sometimes the plans are not updated at all, eventually resulting in "management by crisis" [Kliem and Ludin, 1992], where the project reels from one issue to the next with very little strategic planning being employed.

### 2.2.2 Alternatives to Traditional Project Planning

Obviously some remedy is needed to this dire-sounding situation. A number of alternative strategies have been suggested to combat the problems encountered in traditional project planning. One is to keep the process the same, but acknowledge the time-consuming nature of the updates procedure and bring in special resource to deal with project administration issues. This is an approach known as "The Project Office" [Block and Frame, 1998], where a specialist organisation well versed in project

management techniques and tools handles these aspects of project management. Although this theory experienced some popularity in the nineties, in most industries the pressure to reduce perceived resource "waste" means that it is unlikely to be a viable solution for most organisations. Additionally, the increased overhead to supply the external organisation with the information required to plan may outweigh any savings.

A more fundamentally different alternative is Goal-Directed Planning [Anderson et al., 1988], which aims to focus the project effort against the PSO (People, Systems & Organisation) goals which make the project worthwhile. Rather than planning tasks in detail at the outset, the approach recommends that only an outline plan is produced initially. This skeleton plan should directly relate to the goals of the project. Subsequently, activity plans are constructed using a "rolling wave" approach. This essentially means that detailed plans are only constructed for a phase when it is soon to begin. This increases the chances that enough will be known to create a useful detailed plan. Although the Goal-Directed approach at some stages seems to employ more of a Plan-As-Communication approach, the final products are still high-precision activity plans, so the actual plans in use by those doing the work will reflect the Plan-As-Program mentality.

Milestone Planning[Anderson, 1996] is a method based on Goal-Directed Planning that embraces the Plan-As-Communication model more enthusiastically. It advises against activity planning (an inherently Plan-As-Program approach) since this focuses the effort on time spent engaging in a task, rather than the results to be achieved. Anderson [1996] points out that it is not only difficult to understand the whole at the start of the project, but also unwise to do so, since the results of the earlier stages of the project may lead to unforeseen opportunities or the need for extra effort if things have gone badly. Additionally, the path to the result is left flexible so the most appropriate route can be taken when the time comes.

There are two approaches to scheduling for Milestone Planning. The first is to base performance on past projects and so set the milestone dates accordingly – this is a popular choice if there is some fixed end-point to the project, although schedule pressure and "Can Do" management [DeMarco, 1997, 2002] need to be avoided to keep these realistic. The second is to estimate by identifying the most time-consuming task to complete a milestone and then adding contingency. Although neither of these methods is likely to result in a correct schedule, there is a definite advantage that this is more obvious in this case. Whereas in traditional project planning schedules are often wildly incorrect but look detailed and therefore believable, with milestone planning there is less tendency to believe in unrealistic delivery dates.

Milestone Planning produces low- to medium-precision plans which are much easier to modify. They also follow the Plan-As-Communication model much more consistently, focusing on the results to be achieved and leaving the choice of specific action up to the user of the plan. For these reasons it seems a much more attractive candidate for flexible, reactive planning than the traditional project management approach or the alternatives discussed above.

Despite its apparent attractiveness, however, Milestone Planning has not been adopted by project managers. As we will discuss in the following section, one reason for this may be that the form that the plans take does not lend itself to use by project teams.

## 2.3 Plan Representations

There are a variety of plans produced by various stages of the planning processes already described. Some show just the work to be done and how it can be broken down (e.g. Work Breakdown Structure), others focus on what needs to be achieved (e.g. Milestone Plan) or the dependencies between tasks (e.g. Network Diagram). However, the most popular type of plan currently used combines a calendar/time element with a task listing – the ubiquitous Gantt chart. In this section we will examine the various plan representations in a little more detail.

### 2.3.1 Traditional Plans

#### 2.3.1.1 Work Breakdown Structure

A Work Breakdown Structure (WBS) is a hierarchical list of tasks that make up the overall project. In order to construct the WBS, the external deliverables are considered and the tasks needed to produce them are listed. Then each task is further subdivided until it is of a "manageable" expected duration (heuristics vary: anything from 1 day to 8 week units are suggested). The representation of the Work Breakdown Structure is usually like a family-tree diagram, with child tasks connected to their parent tasks, rolling up into the total project.

WBSs can be useful since they encourage thought about what is actually involved in achieving a task, but they are usually used as a stepping stone towards a Gantt chart, CPS or Network Diagram rather than plans in their own right, since they are really just diagrammatic task lists.

| | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| | **Background** | **9 days** | **Mon 11/10/04** | **Thu 21/10/04** |
| | Background Reading | 6 days | Mon 11/10/04 | Mon 18/10/04 |
| | Project Proposal Preparation | 2 days | Tue 19/10/04 | Wed 20/10/04 |
| | Project Proposal Due | 1 day | Thu 21/10/04 | Thu 21/10/04 |
| | **Literature Review** | **40 days** | **Tue 19/10/04** | **Mon 13/12/04** |
| | Base Reading of Identified Sources | 10 days | Tue 19/10/04 | Mon 01/11/04 |
| | Identification of Specialist Sources | 5 days | Tue 02/11/04 | Mon 08/11/04 |
| | Processing of Specialist Sources | 5 days | Tue 09/11/04 | Mon 15/11/04 |
| | Analysis of Content | 20 days | Tue 19/10/04 | Mon 15/11/04 |
| | Finalise Literature Review | 2 days | Tue 16/11/04 | Wed 17/11/04 |
| | Literature Review Due | 1 day | Mon 13/12/04 | Mon 13/12/04 |
| | **Identification/Development of Methodology** | **14 days** | **Thu 18/11/04** | **Tue 07/12/04** |
| | Write up conclusions based on Lit Review | 3 days | Thu 18/11/04 | Mon 22/11/04 |
| | Identify further research needed | 2 days | Tue 23/11/04 | Wed 24/11/04 |
| | Carry out additional research | 5 days | Thu 25/11/04 | Wed 01/12/04 |
| | Decide upon methodology to support | 2 days | Thu 02/12/04 | Fri 03/12/04 |
| | Write up research & conclusions | 2 days | Mon 06/12/04 | Tue 07/12/04 |

Figure 2.1: Work Breakdown Structure Example – subsection of a larger plan

#### 2.3.1.2 Network Diagram

Network diagrams take many forms, but typically they show tasks, their expected durations and the dependencies between them. Variations exist that show "activity

on node" or "activity on arrow" for instance, but in terms of information presented there is very little difference. In practice, network diagrams are often more useful to illustrate dependencies between tasks as they are being drawn (or to highlight the effect of a delay on the schedule) than as total representations of plans as they can be quite confusing to read.

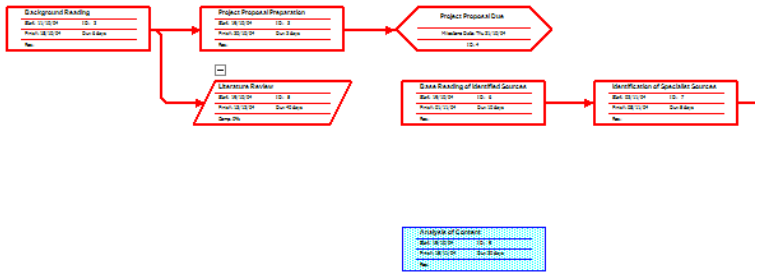

Figure 2.2: Network Diagram Example – subsection of a larger plan

#### 2.3.1.3 Gantt Chart

Gantt charts are a specialised form of bar chart that typically show the task list in rows on the left-hand side and some time line in columns. Rectangles are drawn to show task durations, as well as slack time and dependencies. Although network diagrams are often used to compute dependencies and the schedule basics, Gantt charts are much better visual aids and are better at communicating a plan to project teams or stakeholders[Lock, 2001]. Gantt charts also often have a resource dimension, either showing the person assigned to the task alongside the task name, or occasionally by colouring the task blocks appropriately.

These are often the most popular plan representation, because they are easy to read and give a "big picture" view of the project. The level of precision is very high with this type of plan, because not only are detailed tasks shown, but also the specific dates when they should be worked on and who should be working on them. It is no wonder that in many projects they are left unchanged as the project progresses, as updating all these dimensions would be a time-consuming task.

#### 2.3.1.4 Extensions to Traditional Plans

There are some common extensions to traditional plans, the most common being Critical Paths and Resource Allocation information. Critical Paths are the longest possible paths from the start of the project to the conclusion [Kliem and Ludin, 1993], if you ignore slack. They are usually identified by doing a "forward pass" (calculating the length of the path using the earliest start and finish dates, starting at project commencement) and then a "backward pass" (calculating the length using the latest start and finish dates, starting at the project conclusion and working backwards). The paths for which the total length is the same for each pass are

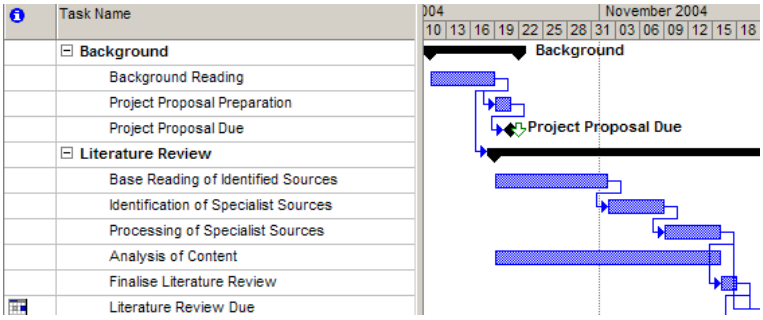


Figure 2.3: Gantt Chart Example – subsection of a larger plan

identified. These paths then represent the collection of tasks that have no slack – no room to manoeuvre.

Resource Allocation information is used to track how effectively particular project resources are being used (or if they are over-subscribed!). It is usually added to Gantt charts or other schedules, showing which team members are allocated to which tasks. This information can then be used for resource levelling, but in practice the degree of effort, estimate detail and accuracy needed to return the correct result is a barrier to this practice. Occasionally the software will try to level resources automatically, but this can be disastrous in knowledge work where resources are simply not interchangeable[Spolsky, 2005].

A variation on the resource angle is responsibility information. Sometimes plans will just show the resource assigned to lead the work on the task, indicating that they are responsible for delivering it. Occasionally more detail is provided, for instance to show the RASCI[1] roles[Quik, 2005]. Responsibility charts can be useful if they are actively used by the project team, but often the responsibilities depicted are not perceived as the "real" responsibilities and so their usefulness is reduced. To be an effective tool, the team processes have to support their use and transparency.

### 2.3.2 Alternative Representations

#### 2.3.2.1 Milestone Plan

A Milestone Plan can have a variety of forms. Those described by Anderson [1996] resemble network diagrams, in that the milestones are represented as graph nodes and dependencies denoted by lines and arrows. The low-precision project maps described by Cockburn [2002a] appear to be a variation of milestone plans as well.

[1] RASCI is a tool used to define involvement in decisions in project management. It is an acronym, standing for Responsible, Accountable, Support, Consult, Inform, defining the roles that various individuals have in a decision or deliverable.

This is best illustrated with an example: *Jack may be responsible for delivering a piece of functionality, but his boss Mary is accountable. Coworkers Jim & Nicola will provide support, helping Jack write the functionality, whereas Rob has experience in the area and so is down as someone to consult. Ellen is working on an associated area and so needs to be informed of the implementation decisions taken.*

In common with Milestone Planning, the key advantage of this type of plan is that it focuses on the results that need to be delivered. Since the plans are low- to medium-precision, they are less fragile than some of the higher precision plan representations (such as network diagrams and Gantt charts). They also do a good job of giving an overall picture of the project without prematurely committing to arbitrary deadlines – a major risk in schedule-pressured projects [DeMarco, 2002, Anderson et al., 1988] – and allow more collaborative thinking about the project plan as a result.

This type of plan is one of the only found that is designed to be a Plan-As-Communication. Although the results that need to be delivered are very clearly depicted, the method of achieving this is not specified. The authors of the approach do recommend, however, that detailed task planning is carried out, so the Milestone Plans are supplemented with traditional Work Breakdown Structures and Gantt Charts.

## 2.4 Planning in Artificial Intelligence

Planning has long been a key challenge in the field of Artificial Intelligence (AI). The ability to plan is seen as a prerequisite for autonomous and intelligent behaviour. We will first discuss classical planning and some traditional methods and then some of the more modern approaches that focus on operating in a dynamic world.

### 2.4.1 Classical Planning

Much of the traditional AI planning research focused on classical planning, where the environment is assumed to be "fully observable, deterministic, finite, static and discrete" [Russell and Norvig, 2003]. This means that the agent can get complete, accurate, up-to-date information about its environment (observable), be sure what the outcome of an action will be (deterministic), comprehend the total environment (finite), assume that only its own actions will affect the state of the environment (static) and that there are only a fixed number of states, actions and so on (discrete).

This information about the state of the environment, actions available, etc, is represented symbolically, so that the agent can reason towards the required goal. Immediately two issues with this approach become apparent, which Wooldridge [2004] describes as the *transduction problem* (translating the real world into a symbolic representation in a timely manner) and the *representation/reasoning problem* (appropriate symbolic representation of information and adequate reasoning using this representation). Essentially, it is difficult to turn the outside world into an internal model and then to plan with it quickly enough to be useful. Although various techniques are used to mitigate these issues, they are fundamental challenges with planning for AI systems generally.

We will now discuss some examples of planning algorithms used in traditional AI planning. The simplest way of thinking of a plan is as a series of actions which will take the environment from the initial stage through to the desired goal stage. When all the possible actions and environmental states are known, it is of course possible to think of this problem as a searching challenge – one need only try every possible

combination of actions until one that results in the goal is discovered. This is know as brute-force searching.

Two planning algorithms that regard planning as search are progression and regression planning [Weld, 1994]. Progression planning is also known as forward-chaining state-space search, because it starts with the initial state of the world and then works towards the final goal. Regression planning is sometimes called backward-chaining state-space search because it does the opposite, starting with the desired goal and working backwards to find a sequence of actions that starts in the initial environmental state. It is easy to imagine that even with a limited number of actions and states, the number of possible combinations (and therefore plans) is immense. Searching through all the possible combinations can be highly inefficient. More recent research (starting with McDermott [1996]) indicates that the use of appropriate heuristics to guide the search can help mitigate this problem, however.

The latter approaches are also sometimes referred to as Total Order Planning, because the resulting plan is a list of actions that have to be executed in exactly the order provided. A more flexible alternative is Partial Order Planning, where the relevant actions that need to be taken are specified in the resulting plan, but order of execution is only specified when necessary. The resulting plan is more a graph of sub-plans than a fixed list, which allows the planner to take advantage of problem decomposition. This type of planning is also sometimes called "least commitment planning" [Weld, 1994], because it allows decisions to be delayed unless there is a compelling reason to decide immediately: for instance, when dressing in the morning, it is possible to delay choosing which tie to wear until after one has put on a shirt; equally, the decision could be taken the night before if desired.

There are of course other classical planning approaches, the most notable omission being a discussion of logic-based planning methods. However suitable logic-based and other similarly mathematical methods may be for computers, they are very unlikely to be useful for planning carried out by business people, who prefer less formal methods. For this reason they have not been included in the discussion here, but for further reading of the area, Wooldridge [2004] has a good overview and pointers to key sources, as do Russell and Norvig [2003].

### 2.4.2 Planning for the "Real World"

Whichever planning method is actually chosen, classical planning methods all have something in common – the plan is produced and then followed exactly, with little regard for changing conditions that might require different action from that which was planned. In theory this is fine, since the approaches described above require the environment to be bounded and predictable in various ways. However, when trying to create agents to act in real-time, in the real world, this kind of simplification is a real barrier to efficacy.

In order to deal with the dynamic, non-deterministic nature of reality, we must build robustness into our planning methods. We have to decide what to do when something goes wrong (e.g. replan or swap in an alternative plan) and also cope with scheduling. One way to do the latter is to separate planning from scheduling

– identifying first what should be done and then when particular steps should be executed. A common framework for doing this is Hierarchical Task Network (HTN) Planning, followed by Critical Path Scheduling. Both these approaches may sound very familiar, since they are very similar to project planning techniques, namely Work Breakdown Structuring and Critical Path Scheduling.

Hierarchical Task Network (HTN) Planning's key principle is to break down the problem into several parts, eventually creating a hierarchy of tasks via action decomposition. Action decomposition is the process of replacing high-level actions with a partially ordered set of lower-level actions [Russell and Norvig, 2003]. The initial plan is thus a high level description of the problem to be addressed, which is successively refined until only primitive actions are left. The key question in this method is how the replacements are found – usually a plan library contains ready-made action decompositions from which to choose.

This raises another issue: how are the plan libraries constructed? Ideally when the agent needs to solve a problem "from scratch", then the resulting plan is stored in such a way that it can be found and reapplied to similar problems. The reality is quite often, however, that human intervention is needed to populate plan libraries for use. As Bryson and Stein [2001] note,"human designers do most of the hard work in artificial intelligence."

Critical Path Scheduling can be applied once a partially ordered plan has been produced – various routes forward and backward through the plan are discovered. The critical path is the longest route through the steps in either direction – a path that consists entirely of tasks that cannot slip (i.e. have no slack). The scheduler then prioritises those tasks that are on the critical path, both in terms of time and resources. The intention is thus that the overall plan will complete in the shortest time possible.

Although HTN Planning does give us the framework to react to changes in the environment, the precise mechanism for reacting to non-determinism in the environment is open to discussion. Next we consider some common options:

- *Sensorless Planning* – systems that use sensorless planning do not identify changes in the environment, but rather construct a plan that will achieve the goal from any initial environment state. This type of planning is also sometimes referred to as Universal Planning [Schoppers, 1987] and relies on the ability to coerce the environment into the desired state. Although in simple cases this can work, in situations where circumstances may well be very unpredictable, its success is unlikely.

- *Conditional/Contingency Planning* – plans are constructed with various branches to cover different eventualities. During execution, changes in the environment are sensed and the appropriate plan branch "swapped in" to amend behaviour accordingly. The obvious flaw with this method is that unexpected circumstances will have no corresponding plan branch and so the agent will either stop or possibly continue executing a potentially ineffective plan. It is also often difficult to identify when a contingency should be sought and also to match the required plan branch to the situation.

- *Execution Monitoring and Replanning* – as the plan is being executed, the environment is constantly monitored. The system responds to change by re-planning. The danger is that in a frequently changing environment, the system may keep replanning and fail to act. Valid heuristics to decide when to act, when to replan and when a plan is "good enough" are needed for this to be an effective planning method. This need has led to the development of Anytime Algorithms [Zilberstein, 1996] which always return a useful/correct plan, but produce greater quality plans the more time allowed for processing.

Another school of thought focuses on reactive and hybrid approaches. One of the commonalities of these various approaches is the belief that "intelligent behaviour emerges from the interaction of various simpler behaviours" [Wooldridge, 2004, p.89]. Grand [2001] took this concept to the extreme with his work in the area of artificial life, producing seemingly intelligent creatures by modelling physical entities (including brain synapses and hormones) to build up a complex being. The more familiar model used in reactive agents, however, is the subsumption architecture proposed by Brooks [1986]. This multi-tiered architecture offers a number of simple behaviours, organised in a hierarchy – these are matched to the perception of the environment and called appropriately.

Surprisingly complex emergent behaviour is possible using this type of approach. This was originally demonstrated by Agre and Chapman [1987] with their PENGI system. Their agent played a computer game called PENGU and exhibited what seemed to be rational, planned behaviour when in fact it was merely encoded with some simple opportunistic behaviours. This type of approach seems the antithesis to the planning already discussed, but in fact there is a great deal of study devoted to "reactive planning".

> "Reactive planning is something of an oxymoron. It describes the way reactive systems handle the problem traditionally addressed by conventional planning: action selection. Action selection is the ongoing problem (for an autonomous agent) of deciding what to do next" [Bryson and Stein, 2001]

Reactive planning differs from classical planning in a key way. Whereas classical planning maps an entire series of actions, starting at the initial environmental state and ending in achievement of the goal, reactive planning is only concerned with the immediate next action. Reactive plans are therefore constructs which specify the action to be taken next, given the current state of affairs. These are sometimes also known as condition-action rules [Pryor and Collins, 1996]. In the following section we will discuss how this and various other AI planning approaches might be reapplied to the project planning domain.

## 2.5 Discussion :: Parallels Between AI and Project Planning

There are a number of parallels between artificial intelligence planning and project planning. There are also some clear divergences. Our aim in this section is to

discuss the similarities and differences between AI and project planning, identifying both avenues that have already been explored and possibilities yet to be tested.

Most striking in the AI planning world is the plethora of approaches, both for tackling classical planning problems and the issues of operating "in the real world". In project planning, on the other hand, relatively few alternatives have been proposed, with most writers in the field seeming to believe that the traditional project planning process is the ideal.

Although a few bravely try to suggest that some aspects of the traditional project management process are difficult to follow for the busy project team [Anderson et al., 1988, DeMarco, 1997], the majority choose to believe that the process is simply not being followed well enough. Many more stringent methods are suggested, from improving estimation using scientific methods [Goodpasture, 2004] through to having a specific group of people devoted to planning, separated from the project [Block and Frame, 1998].

The picture in AI planning research is very different. A number of methods have been tried and proven not to be satisfactory, leading to the development of other approaches and their testing, in turn. Where the reasons for a method being flawed are transferable (i.e. still relevant when planning and execution is being done by humans rather than machines), the lessons learned are potentially very interesting to reapply to project planning.

Let us first consider classical AI planning. The attitude to the environment and to the plans themselves is very similar to the traditional project management approach. External events that might influence the environment are no more considered in traditional project management than in classical AI planning. Both appear to believe that if the initial state, actions available and desired goal state are known, then a plan can be constructed. That plan can then be followed exactly and at the end the goal state will be reached. Lock [2001], however, regards this blind following of a plan rather disparagingly:

> "There is an alternative management approach that relies only on outgoing instructions, with no feedback or error signals. This is called "management by surprise" because the manager feeds in work at one end of the system and is surprised when it doesn't come out the other!" [p.226]

One definite difference between classical AI approaches (such as progression & regression planning) and traditional project planning is the approach to task selection. Whereas the aforementioned computerised methods consider all possible task combinations and then search for those sequences that begin in the right state and end in the goal state, human beings employ filtering in selecting tasks. This is both logical (knowing, for instance, that shoes should not be put on before socks) and sometimes based on previous experience (e.g. knowing that writing a long document will take at least three times as long as originally estimated).

Humans have an intrinsic ability to consider only the most relevant potential actions. They are also adept at learning from experience and applying this knowledge to situations identified as similar. An example of this type of filtering is discussed by

Caulfield and Bryson, when investigating how humans choose their next move when playing chess. People are also good at identifying cases where the order of action does not matter, so although Partial Order Planning was a great breakthrough for AI planning, it would not be a new concept to project planning. Although some plan representations fail to show the non-sequential nature of some tasks, the most popular – the Gantt chart – is perfectly suited to do so.

When we consider AI planning methods that have been developed for operation "in the real world", however, more parallels and opportunities become apparent. Hierarchical Task Networks evidently lean heavily on the Work Breakdown Structure plan format that is an integral part of traditional project planning. Sensorless Planning is sadly similar to the "management by surprise" situation described above by Lock [2001], but Conditional/Contingency Planning seems an interesting avenue to explore.

Traditional project management does not identify contingency planning as a needed step, however sensible it may seem to consider situations that might arise and construct alternative plans. The only method that advocates such an approach is in fact a software engineering methodology – Boehm's Spiral Model [Boehm, 1988]. This methodology is focused around risk management when building software : after each iteration of gathering requirements, designing, implementing and testing, there is an explicit risk analysis stage.

Some other project management texts, such as Kliem and Ludin [1993], suggest that risk analysis is an important process, helping reactions to problems to be reasoned and constructive rather than the first solution to come to mind, but most do not cover risk management beyond the initial feasibility study. Even in those cases where risk management is discussed, it is not regarded in the same way as AI's contingency planning. Rather than providing new branches of the plan to replace the existing plan, risk management strategies are usually just measures to overcome the risk, with the project thereafter continuing to follow the original plan.

In fact, the approach that is prescribed by those texts that do recognise that the environment or circumstances may change is most similar to the AI approach of Execution Monitoring & Replanning. One might assume, therefore, that the issues experienced when using this approach in artificial intelligence are overcome by a human planner, in the same way that the action selection problem is. Unfortunately, this is not the case. Although most texts hold that replanning is needed whenever circumstances change, they also bemoan the time that this takes:

> "The project manager becomes sucked into a spiral of planning and replanning. He must assign new roles and responsibilities for new work, while old resources are assigned to new tasks for which they are unsuitable. Eventually, the project manager spends so much time replanning that he ceases to manage" [Anderson et al., 1988, p.16]

One difference is in where the problem lies. At root, the issue is that too much time is spent replanning rather than acting – the result being that when the circumstances are changing frequently, the planner can end up failing to act completely, stuck in a

replanning loop. What tends to take up most of the AI planner's time is choosing the correct course of action – whereas in project planning, human planners tend to update the tasks relatively easily and correctly, but problems occur in another area: estimation.

Estimation is not only a problem when replanning – it is clearly a challenge throughout the planning process, with a great deal of research dedicated to improving estimation methods. Quantitative methods are suggested by Goodpasture [2004], weighted averages by Kliem and Ludin [1993] and rules of thumb by Sommerville [2001]. In practice, however, estimation appears to be the area most skimped upon [Williams]. This is truly dangerous, since a detailed plan with incorrect estimates is often a sign of "Can Do management" [DeMarco, 2002] – and indistinguishable to stakeholders from a real plan. This, combined with inadequate tracking and reporting, can lead to runaway projects that go over schedule and budget without the stakeholders realising.

An approach that has little reflection in project planning methodology is Reactive Planning. This seems counter-intuitive, since it is the area most focused on reacting to changes in the environment. In some ways, though, it makes sense, since project management is specifically targeted *against* purely reactive behaviour. A situation where the project manager is simply reacting to events in the short term is regarded as the undesirable inverse to a mediated, long term, planned course of action. Thus, reactivity is frowned upon, as exemplified by Kliem and Ludin [1992, p.77]:

> "Project managers must follow their plans. Little sense exists in investing much time and money in building plans and then never taking time to ensure that everyone (including themselves) follows them. Yet not only do few project managers plan, but they very rarely follow the plans they have developed. They may build an elaborate plan perhaps to prove to their management and themselves that they know what they are doing. Then they quickly deviate from it. The result is project managers with a reactive style of management. The project is replete with examples of management by confusion, management by drives, and management by crisis. Project managers soon ask themselves: What went wrong?"

As we have already discussed, however, the advocated "execution monitoring & replanning" approach is deeply flawed, because too much time is spent adjusting the plans rather than actually modifying the actions. So perhaps there are lessons to be learned from Reactive Planning? The first to come to mind is the shorter term focus. As previously stated, the key task of a reactive planner is selecting the immediate next action, taking into account the long term plan, the environment and state and the actions available.

The only similarity that could be found in project management is the "rolling wave" approach advocated by Anderson et al. [1988], where detailed planning is only attempted when the relevant section of the project is about to be undertaken. This does not preclude longer term, less detailed planning (which is indeed also the case in Reactive Planning), but allows the focus on specific actions to be taken only when appropriate. This has the added benefit that the overall plan does not have arbitrary

detail in areas that are far into the future and therefore less likely to be correctly planned.

We also again have an analogy to risk management – the condition-action pairs that are a key structure in reactive plans are similar to risk plans, where a potential risk is described and the actions to be taken to mitigate it if it occurs. In fact risk management is often dealt with by managers in a very reactive manner – when a risk scenario plays out, the risk plan is seldom fed into the overall project plan, but instead followed outside of the normal course of operation to deal with the risk.

Arguably direct reapplication of reactive planning would be difficult, since a great deal of skill is required to translate reactive behaviour (i.e. choosing the immediate next action) into goal attainment (i.e. achieving the end-state actually desired). However, this does seem the most interesting area to explore in order to develop a method for dynamic and flexible project planning. The eventual focus on the goals to be attained and the behaviour that emerges from the simple selection of next-actions could prove to be highly useful in focusing project teams on project results rather than day to day tasks.

Having considered traditional project planning and the existing alternatives, we have found that there is no successful methodology designed for flexible project planning. We therefore turned to the area of Artificial Intelligence planning, considering traditional approaches, finding that these were similarly flawed. In examining more recent advances in AI planning, however, we have found some interesting concepts, particularly in the area of Reactive Planning. In order to apply these to project planning, however, we must first delve deeper into the way that project plans are actually used, which is the focus of the next chapter.

# Chapter 3

# Dynamic Planning – A New Approach

## 3.1 How Are Plans Really Used?

The various project planning approaches already considered make assumptions about how plans should be used. Some of those that discuss the issues with traditional project planning consider the reality of where this process falls down, but almost all assume that the root issue is lack of adherence to the advocated process, rather than the process itself.

In AI planning, however, because the planning methods are being followed by agents rather than people, it is easier to measure where the process itself may be the problem. Once programmed with a certain approach, absolute adherence can be expected from agents and indeed it is possible to "step through" the planning process and see the plan being constructed and followed[1].

As indicated when discussing the various AI planning approaches, issues and limitations have been investigated and identified. Why then is this not the case in project planning? In fact some sources (for instance Chatzoglou and Macaulay [1996]) have examined the limitations of various methodologies. DeMarco [1997] also discusses the typical failings of the project planning approaches. Most of the research in the area focuses on problems with the planning process. Little or no research appears to discuss how the plans produced by the existing approaches are actually followed.

Although the simple act of constructing a plan can be very useful to gain a fuller understanding of the project, an idea of which areas will be especially challenging and where potential resources constraints will come to light, most project management texts intend the plans to be followed. As we heard from Kliem and Ludin [1992] earlier, project managers must follow their plans and ensure that the team does as well.

[1] Testing in this way (where the inner workings are examined) is known in software engineering as White Box Testing [Sommerville, 2001].

### 3.1.1 The Direct Research

In order to investigate whether this actually happened in practice, a number of plan users were surveyed. On the one hand, a group of Information Technology managers from a large multinational corporation were approached – let us refer to this as Group A. Due to the nature of the corporation and its training strategy, these individuals were all likely to have taken very similar project management training. They were also at different stages in their development as project managers, from relative novices in charge of smaller scale projects through to large regional or global projects. The expectation was, therefore, that they would use similar plans and planning techniques, but apply them to projects of various levels of complexity and in various stages of maturity.

The second group (Group B) was a more random sample, consisting of individuals who responded to a call for respondents by the author, either in person or on the Internet. This group included students, technical managers at various firms and lecturers. The expectation was that this group would be involved in diverse types of projects, although all would be related to technology in some way (due to the recruitment method). Equally it could be expected that the individuals in this group would not all follow the same method for project planning or use the same type of representations.

All respondents were asked to describe the planning process they typically used, as well as the plans produced, including samples where possible. Follow-up interviews were held to pursue particular points and to clarify assumptions. Although the sample size was relatively small (a total of 12 respondents, split equally between the groups), the open-ended nature of the questions meant that a great deal of interesting qualitative data was gathered.

### 3.1.2 Key Results

There were three general plan types that occurred again and again in the descriptions and samples of the respondents. The most popular (which 92% of respondents had) was a Gantt chart style plan. This was typically used for broad project team and stakeholder communication, often with the same plan being used for both groups. Most of the plans consisted primarily of tasks, showing duration, start date and end date for each.

However, on a number of plans milestones were also identified and almost all of the plans that were intended for group use also exhibited some form of resource allocation. This varied from the very simple (initials on the task bars) through to the comprehensive & complex (columns corresponding to the RASCI role assignment model previously mentioned).

The most interesting thing about the Gantt charts was not the content, but the users' attitudes and comments about their use:

> "But to be honest, this is written at the start and then it just merges into my todo list in xls *[Microsoft Excel spreadsheet tool]*" – Respondent A5

"The Gantt chart is out of date!" – Respondent B2

"*[The Gantt charts are]* referred to and updated in fortnightly meetings during the projects" – Respondent A2

"... you need discipline to keep going back to the original plan" – Respondent A4

The impression here is that at best the plans are updated every two weeks and in some cases they are not updated at all! The fact that they are often not referred to except on an infrequent basis strongly suggests that they are not used to decide the day-to-day actions of the team members. Respondent A5 calls this out in the first quote above and the talk of todo lists and action plans by a number of other respondents supports this assumption.

It is not surprising, therefore, that the next plan type was the personal todo list or action plan. The former is the ubiquitous list of tasks that an individual plans to undertake; the latter is similar in structure but usually decided upon during a team meeting. Occasionally action plans are for the entire team (with responsibilities indicated) and so are project rather than individual action plans. The structure of this type of plan again ranged from the extremely simple (the simple list of tasks) through to annotated versions of the todo list, for instance showing deadlines or notes and finally to more complex forms (such as the time management grid shown in Table 3.1).

The third type of plan, which arguably combines the two already mentioned, was the augmented calendar. This type of plan essentially consists of a standard calendar, marked with deadlines, milestones and tasks. An example can be seen in Figure 3.1 on the following page. The key advantage of this type of plan is that it makes it easier to handle multiple projects and priorities – the user can set time aside to work on each. Although these were only explicitly called out by the members of Group B, further investigation found that those in Group A all use a centralised corporate calendar system to book and keep track of meetings, events and appointments. They indicated that because this is standard procedure across the corporation these type of plans are so integral to their daily work that they did not even think to mention them.

| | |
|---|---|
| *Important and Urgent Tasks*<br><br>E.g. Finish write-up | *Important but Not Urgent Tasks*<br><br>E.g. Revise for exams |
| *Urgent but Not Important Tasks*<br><br>E.g. Go to sale | *Not Important and Not Urgent Tasks*<br><br>E.g. File bills away |

Table 3.1: Time Management Grid

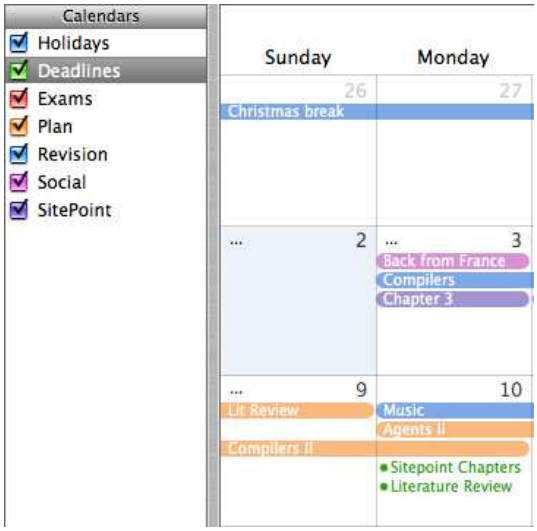


Figure 3.1: Augmented Calendar Example[2]

The key learnings from this investigation can be summarised as follows:

- Most respondents (92%) use a Gantt chart style of plan, but this is often not kept up to date.
- All respondents reported using an additional individual plan, either in the form of a todo list or the hybrid approach of the augmented calendar. Some respondents use both of these in additional to project plans.
- Almost all (11 out of 12) used colour in their plans to add extra information.
- Half reported that their plans included resource allocation (primarily human resources & responsibilities).

## 3.2 What Does This Mean For The Existing Planning Methodologies?

The most interesting finding is that individuals all have personal plans in addition to project plans. This, combined with the comments about the out-of-date nature of most of the project plans, indicates that respondents *do not use the project plans to decide their everyday actions*. Even when they are in control of the overall project plans, users report that they still have personal todo lists and often do not keep the project plans up-to-date.

What does this mean for existing project planning methodologies? If the plans are not being used to decide day-to-day actions of the project team members, then it means that any time spent inserting detail of what actions are to be taken and when is arguably time wasted. Nevertheless, the plans are still used in other ways, so the

challenge is to identify the bare minimum planning that could be done and keep the plans useful.

Another clear message from the Literature Survey and the direct research is that the Gantt chart is a very popular plan representation. Despite the emphasis in both project management and software engineering on network diagrams, none of the respondents (who all work in Information Technology roles) identified network diagrams as a form of plan that they used. In fact, a few respondents identified the dependency visualisation of Gantt charts as relatively useless, citing the cumbersome nature of this functionality in most planning software packages as an irritation.

## 3.3 The Need For A New Approach

So far planning in general may have seemed somewhat disappointing – there appear to be countless problems with the methods used to construct plans, their representation and the way in which they are followed. Nevertheless, planning is essential to ensure that projects achieve their objectives, agents meet their requirements and that both people and agents take the best action to achieve the desired goal.

What has become clear, however, is that there is a definite deficiency in the existing project planning methodologies. We have already seen that some great advances have been made in the field of artificial intelligence planning, though, so perhaps some of this knowledge could be reapplied to the project planning domain.

This is in fact the approach that we will take in the following subsection, but first let us recap the key problems with existing project planning methodologies and the plans they produce:

- Planning is a time-consuming process, which is often neglected .
- Estimation is a key challenge and, when done badly, results in incorrect plans that may nevertheless seem credible.
- A great deal of time and energy is put into planning, but then the project plans are not used as intended.
- Project plans usually consist of tasks which the project team are intended to follow, but in reality most individuals prefer to follow their own todo lists or action plans.
- Project plans are frequently out-of-date, supporting the tendency to plan separately at an individual level.
- Although there are multiple plan formats available, only a few appear to be used.

It is obvious that there are a number of problems in three key areas: the way that the plans are constructed, their format and the way in which they are used. In order to construct a project planning methodology that provides sufficient flexibility to meet the demands of modern projects, all these areas will need to be addressed.

## 3.4 Proposal: Dynamic Project Planning

In defining a new project planning methodology that deals with the aforementioned problems, we will take a somewhat unconventional route, working backwards and starting with the issues in how plans are used, then discussing the format and finally the method of construction. This is so the entire methodology is constructed with the end in mind – providing flexible plans that are an asset to the project team and help project managers react to changing circumstances.

### 3.4.1 How Will The Plans Be Used?

To go back to Agre and Chapman [1989] and their seminal paper "What Are Plans For?", one key distinction that can be drawn between plans is their purpose. On the one hand there is the Plan-As-Program, where the plan provides the exact actions to be taken, the order in which to do so and often a schedule as well to show when tasks should be completed. As we have seen, most AI planning methods provide exactly this type of plan. This is in one way not particularly surprising, since it is well accepted that computers need explicit instructions to perform tasks[3].

What *is* surprising, however, is that most of the project planning methods produce Plans-As-Programs too. Humans are the key example given of autonomous action, of almost subconscious planning to achieve a wide variety of tasks. People have the amazing ability to adapt to totally new conditions, one reason that the human race has spread to every continent and survives in climates of every extreme. We, as a race, also have the enviable ability to learn from one experience and reapply the knowledge to an alternative situation, both identifying the similarity and adapting the original situation to the new circumstance.

All these things combine to make it seem strange that plans for use by humans are Plan-As-Program, rather than Plan-As-Communication, where the plan is used to communicate the goals to be achieved in a far less specific way. Nevertheless, this seems to be the standard. What we have already seen when asking people how they use project plans though is that the prescription of activities is ignored. People seem to recognise in themselves the ability to decide exactly what to do for themselves and to do so regardless of the level of detail supplied in the plan.

One explanation of why this is so comes from the area of management study referred to as Organisational Behaviour, which focuses on internal dynamics in companies. A revolutionary book by McGregor [1960] challenged the core assumptions of the scientific approach to management, begun by Frederic Taylor in the 1890s. He referred to the existing approach as "Theory X" and to his alternative as "Theory Y". The key attributes of each approach can be seen in Table 3.2 on the next page, which was constructed from information in Buchanan and Huczynski [2003].

Despite the fact that most modern organisations subscribe to McGregor's "Theory Y", rather than the more traditional approach that demands employees be told exactly

[3] Some new approaches, such as the Artificial Work typified by Grand [2001], take the alternative approach that software agents can be built up with the inherent ability to take decisions and perform tasks, but the programmed method is still very much the standard.

what to do, the plans used in such organisations do not reflect this shift at all. Despite corporate policy of distributing authority and responsibility, encouraging employees to take pride in their work and to be creative, project plans are still full of tasks, telling employees what to do and in what order to do so.

| Theory X | Theory Y |
|---|---|
| Average person inherently dislikes work | Work is as natural as rest to people |
| People must be directed to work | People will exercise self-discipline and self-control |
| People wish to avoid responsibility | People enjoy real responsibility |
| People feel achievement at work is irrelevant | Achievement is highly valued by people |
| Most people are dull and uncreative | Most people have imagination and creativity |
| Money is the only real reason for working | Money is only one benefit from work |
| People lack the desire to improve their quality of life | People have needs to improve their quality of life |
| Having an objective is a form of imprisonment | Objectives are welcomed as an aid to effectiveness |

Table 3.2: Key Tenets of Theories X and Y

So how should plans be used? Well, since a number of those interviewed are very successful, one can assume that the way that they use plans is valid. We should expect project team members to supplement project plans with their own personal plans, in the form of todo lists, or action plans, or augmented calendars. The key need therefore is for the project plan to communicate the objectives, priorities and immediate goals of the project.

It is important, at this stage, to note that different roles require different things from the project plan. The project manager needs to use the project plan to track progress, so that adjustments and interventions can be made. They also need to be able to plan key resources, at least in the short- to medium-term. Other stakeholders may need a high-level view of the project progress to date and upcoming work. Although the Systems-Gap-Working-Party [1984] found that a key issue was "one size fits all" plans and that differing plans, with differing levels of detail, should be constructed as appropriate for the various stakeholders, this either needs to be achieved without a great deal of extra effort, or the standard plan needs to meet at least the basic needs of all involved.

Based on the indications thus far, we will assume that task planning can be discarded and that the following are the key ways in which plans will be used:

- Project team members will use the plans
  - to identify current objectives, priorities and short-term goals
  - to source their individual task/action plans

- Project managers will use the plans
  - to map out the objectives, priorities and goals of the project
  - to track project progress and identify adjustments/interventions to be made
  - to allocate resources and identify any resource conflicts

Essentially we are reapplying the principles of the Reactive Planning AI approach, but relying on the project team members to make the appropriate action selection based on the context and environment. This allows the project manager to focus the plan on the changes to be achieved with the project (just as reactive agent programmers focus on the emergent behaviour desired from the agents). It also allows a more motivational working environment for the team members, since this approach will allow their creativity and autonomy to shine through and fuller responsibility and authority to be distributed to each team member.

You will notice that we have excluded the way that other stakeholders will use the project plan. This is both to narrow the focus and based on the assumption that the project plan may not be the best communicative tool for stakeholders outside of the immediate project team. This is arguably an area that deserves further investigation at a later stage, however.

### 3.4.2 What Will The Plans Look Like?

Now that we have set out the key principles and assumptions as to how the plans will be used, we can focus on what the plans themselves will look like. Although we reviewed a number of different plan representations during the Literature Survey, it became apparent during the research that very few of these are used in practice. Although it could be argued that this is because most of the plan representations are not fully understood, for our purposes here we will assume that the most prevalent plan type is popular because of its utility and simplicity.

The most popular is of course the Gantt chart. Its simplicity and graphical nature allow for "at a glance" understanding of the project, or the work currently being undertaken, but it can also be augmented with a great richness of detail. As we saw during the direct research, no single standard format is used across the board. The key features of the Gantt chart are bars to show activities, with their length indicating their duration (since the x-axis shows a time scale). Colour-coding is used in various different ways as well, the only real standard being its presence – even plans constructed by a blind user who participated in the study included colour-coding to denote project phases!

Based on the number of variations discovered even when surveying only a small group, the key requirement thus appears to be flexibility in the presentation of the plans. Thus, the plans will be loosely based on Gantt charts, but the primary feature of the plans will be that they allow the users flexibility in how they choose to denote responsibility, resource allocation, progress and so on.

The traditional Gantt chart does have a key disadvantage however – the basic unit is the task. Since we have already decided that plans which lay out exactly what needs to be done and when are a) not used as intended and b) not designed to take advantage of human skills, a plan that consists completely of tasks and the times when they should to be performed is obviously not ideal.

Other researchers have identified that activity planning is often not useful [Anderson, 1996], but usually with a focus on the inability to correctly plan detailed activities at the beginning of the project. Anderson, Grude, Haug, and Turner [1988] earlier suggested Goal-Directed Planning that focuses on what the project is aiming to achieve and a more recent paper by Anderson [1996] suggests Milestone Planning instead. The key issue with both of these approaches is that they produce plans that are much more akin to the seldom-used network diagram than the ubiquitous Gantt chart. Additionally, they propose this milestone-focused planning only at the beginning of a project, advocating detailed activity planning in a "rolling wave" approach.

We propose a much more radical approach, combining the Milestone Planning approach with AI's Reactive Planning. In our plans, there will be no tasks, only milestones. Task planning is left to individuals and left completely out of the overall project plans. We also incorporate the idea of miniature milestones proposed by McConnell [1996] (sometimes also known as "inch pebbles"). These allow milestones to be divided into submilestones and these in turn to miniature milestones.

Ideally a miniature milestone should signify a result achievable within a week. This allows the project manager to easily track progress, in fact much more easily than was possible with traditional task-based plans, since a milestone is either achieved or not. They are binary measures of progress, rather than relative measures – tasks can be 10% complete or 90% complete and the measurement relies completely upon estimation, a generally flawed activity (as we have seen).

There are a number of advantages to this approach. It focuses the team on what is meant to be achieved in the project, rather than the activities that they will engage in to achieve the end result. This should lead to more aggressive pursuit of results, rather than "spending time" working on the allocated activities at a particular time. Milestones are also less likely to change than activities. This means that the project plan will be more robust and the reactivity to the changing environment left in the hands of the person responsible for delivering the milestone, arguably the best equipped to detect and respond to changes in situation as they occur. The easy progress monitoring will also help the project manager identify when extra investigation or intervention is needed.

The one remaining problem is the reconciliation with the representation of milestones in a Gantt chart style. Milestones are typically represented as a circle with some symbol that is linked to its description. As mentioned, Milestone Plans are usually more like network diagrams than anything else. A key advantage of the Gantt chart is also that the bars give some indication of duration, which helps to identify any resource conflicts.

In order to solve this dilemma, we will represent milestones as bars, with the right-hand edge of the bar representing the time period in which the milestone will be

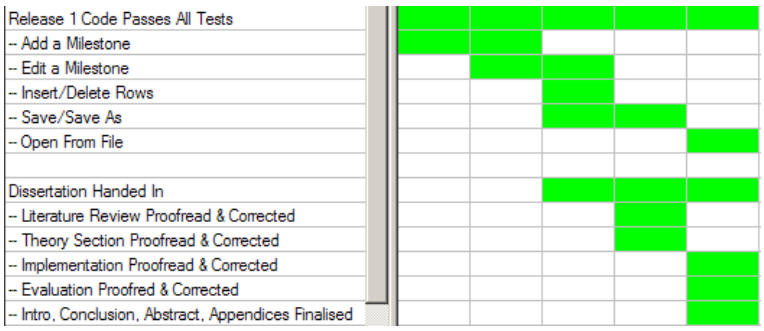


Figure 3.2: Dynamic Plan Example (medium-precision)

completed and the length of the bar giving some indication of the amount of effort expected. Another advantage of this approach is that it is conducive to planning at various levels of precision. In the initial planning stages, a milestone's due date may be expressed in terms of a particular quarter or even year. Later on it may be tied down to a particular month or week. An example of a medium-precision dynamic plan can be found in Figure 3.2.

To summarise, the key features of our plans are as follows:

- The basic unit of the plan is the milestone, shown as a Gantt chart bar, with a due date and an effort estimation to show when resource is likely to be needed against achieving that milestone
- Milestones contain submilestones and miniature milestones to aid progress tracking
- Colour-coding and milestone labelling (as text alongside the milestone description) are both available to be used by the user as they wish (e.g. for resource allocation or tracking delays)

### 3.4.3 How Will The Plans Be Constructed?

In discussing the plan that is the output of the planning process, we have of course discussed some aspects of how the plans are constructed. Therefore here we will merely summarise the process.

At the beginning of the project, a low precision plan will be constructed, to communicate to the project team the key aims and objectives of the project. This will be done by dividing the end result of the project into a number of coarse milestones, identifying the key deliverables that will add up to the desired product or change.

The medium precision plan needed later on will be constructed by dividing each milestone into a series of submilestones, adding resource allocations and any other information that is needed. Higher precision plans will incorporate miniature milestones as well, but these should only be constructed for the immediate next phase

of the project. This avoids planning detail far into the future, when it may not even been needed or issues are not well understood.

During the original background research for this project, various existing planning packages were reviewed. The details of the review can be found in the appendices. The net outcome, however, was that all existing software was very much geared for task-based Gantt charts – to the extent that it proved impossible to attempt to construct a milestone plan, or the dynamic plans suggested here, using them. For this reason, it was necessary to develop software specifically designed to support Dynamic Planning, in order to evaluate the methodology. In the next chapter we will discuss this process and the application produced.

# Chapter 4

# Development of The Dynamic Planner

To facilitate the evaluation of the Dynamic Planning approach, it was necessary to build software that would support planning in this way and produce the type of plan specified in the previous section. Here we will document the process followed in developing this software, as well as discussion of key decisions that were taken.

## 4.1 Approach to Development

The challenge was to produce a useful piece of software in a relatively short space of time. For this reason, a software development process that supports rapid, evolutionary prototyping was needed. An iterative approach seemed prudent and it was also decided that having fully functional software at the end of any given iteration would be desirable, in case any risk factors presented themselves and cut development short.

For all these reasons an Extreme Programming approach was chosen, as described by Beck [2004]. The key advantages of Extreme Programming are its focus on simplicity and elegant design, working code at every stage and a steady, predictable progression from one release to the next. Some aspects of the process could not be followed, most notably the *Pair Programming* and *Customer Always Available* tenets : the former was not possible because the assessment for this project must be individual and the latter difficult because there was no real customer for the project.

## 4.2 Technology Selection

The need for rapid development was again a significant factor when an appropriate technology needed to be selected. So was the nature of the application – since the intention was for it to help the user construct a project plan, a graphical user interface (GUI) was required and so the availability and quality of GUI toolkits also needed to be considered.

Another key decision was whether the application should be web- or desktop-based. Having reviewed a similar project undertaken by a previous student in 2003/4 [Richards, 2004], where the implementation was web-based, the usability issues presented by having to reload the page every time information was stored or loaded seemed to advise against the development of a web-based system[1].

Thus, the list of potential technologies was reduced to those capable of desktop-based GUI application development. The key options in this area were traditional languages such as C++ and Java and modern scripting languages such as Python and Jython.

Both C++ and Java were strong contenders due to their object-orientation support and extensive GUI libraries. Although C++ has no native GUI libraries, the wxWidgets/wxWindows toolkit [wxWidgets Website] is a freely available resource that provides all manner of "native-style" widgets for GUI development on various platforms. The Java Swing/AWT toolkits [Java Website] have their own distinctive look & feel, but since Java itself is capable of running on any platform that supports the Java Virtual Machine, this option is also platform-independent.

As already mentioned, however, rapid development was the key aim and for this scripting languages clearly excel. Their reduced formality and high-level functionality helps the developer to code quickly. Since less code is typically produced by a scripting language compared to the more traditional languages already mentioned, some also suggest that debugging is easier [Pilgrim, 2004]. Python provides a choice of GUI toolkit, including Tkinter (the native toolkit which is unfortunately somewhat limited) and wxPython [wxPython Website] – a Python API to the wxWidgets/wxWindows C++ GUI libraries already discussed above. Jython [Jython Website], in turn, provides all the flexibility of scripting language but with full access to all Java's standard libraries – including, of course, the Swing/AWT toolkits.

The choice was thus between Python and Jython. Either choice would require learning a new language, but since Jython essentially uses the Java Swing/AWT GUI toolkits which I had previously used, only Python offered the prospect of a new programming language and a whole new GUI toolkit. Since one of the objectives for the project was to gain new skills, this option seemed more challenging and therefore more desirable.

## 4.3 Requirements Elicitation

### 4.3.1 Process Followed

Since the software was largely developed as a vehicle for testing the Results-Focused Planning methodology, much of the requirements elicitation consisted of inspecting the methodology and identifying the process steps that needed to be supported. This

[1] Although more recently the development of the AJAX (Asynchronous Javascript And XML) technologies [Garrett, 2005] has heralded a new era of fast and responsive web applications, at the time when the technology was chosen for this project, the AJAX approach was relatively unheard of.

was supplemented by evaluating existing planning tools to identify which features worked well, which less so and which were incompatible with the new process.

The key vehicle used to identify requirements from the methodology description were "User Stories" [Beck and Fowler, 2004], the primary building block of Extreme Programming. User stories are a way of describing software features, based on the popular "Use Case" requirements specification method [Cockburn, 2002b]. Whereas the primary focus of standard use cases is to specify the exact behaviour of the system so that development team and customers can agree on what ought to be implemented, the focus of user stories is to divide the functionality into chunks that can be delivered incrementally. The full user stories can be found in the Requirements Document in the appendices.

The requirements elicitation and specification process was well-suited to this project. Since there was only one developer and no real customer for the system, the relative informality of some of the processes did not cause any operational difficulties. Had the software been intended for general release as user software, then to complete it in the same amount of time would have required a larger team, a clearly defined customer or user group and potentially a more rigorous structure to the requirements gathering and documentation, to ensure that all stakeholders were properly informed. Additionally, a more user-centric requirements elicitation process (for instance one of the various options discussed by Preece et al. [2002] ) would have been preferable.

### 4.3.2 Key Requirements

Since only the basic functionality was needed to carry out initial tests into the efficacy of the new methodology, this was the primary focus during the development phase of this project. Nevertheless, the requirements elicitation did extend further, to the intermediate and advanced level requirements which would make the Reactive Planner suitable for use as a standalone piece of user software for project planning. These are documented in the appendices as reference for further work.

The key basic requirements that needed to be satisfied for the purposes of this particular project were as follows:

- Allow construction of a plan completely from milestones
- Display the plan as a Gantt-style chart, showing milestones and estimations of the duration of the work to achieve those milestones
- Allow the user to save the plan, as well as to reopen and edit it

Although these may seem deceptively simple, the resulting application is surprisingly powerful. The approach taken throughout was to allow the user as much flexibility as possible, to extend or adapt the plan as they wished. This resulted in a very useful application, despite the apparently sparse feature set.

## 4.4 System Design

### 4.4.1 High-Level Design

The overall architecture of the system reflects the Model-View-Controller (MVC) pattern [Krasner and Pope, 1988] for user interface design. This pattern separates out the application appearance *(View)*, the application data *(Model)* and the user interaction with the data via the interface *(Controller)*, so that each aspect can be worked on independently. Although this pattern is useful in terms of code division and clean design, it has the added benefit that representations, interactions and data are separable. This means, for instance, that the way the plan is displayed within the application is separated from the actual plan data, so exports to alternative formats should be easily implementable.

The code structure of the application mirrors the MVC pattern well. Corresponding to the View aspect, there is a specification of the structure, components and "look and feel" of the user interface itself. The Controller consists of a number of event listeners and event handler methods – when the user manipulates the interface, then the event listeners detect the user action and call the corresponding event handler method. This, in turn, updates the data (Model) as appropriate. Although the Model code is contained in a separate class file (*planData.py*), the View and Controller code are both contained in one file (*PlannerGUI.py*) due to the way in which the wxPython framework works.

### 4.4.2 User Interface Design

The Graphical User Interface (GUI) was first mocked up in a series of paper prototypes. The starting point was a composite of the various interfaces of the existing planning software packages that were reviewed. Then everything that was not necessarily relevant and required by Dynamic Planning was cut away, including start dates, end dates, resource allocations, completion percentages and so on. From this stark basis, a number of sketched iterations led to a design that was sufficiently advanced to be implemented.

Obviously during the course of development the interface developed further as functionality was added. Throughout, the user interface design principles articulated by Sommerville [2001, p.330] were used as a checklist to identify possible problems with the interface. Additionally, a number of "Quick & Dirty" interface evaluations (as described by Preece et al. [2002, p.341]) were executed with experienced planners. Although these were perhaps not as effective as full user/customer involvement in the design process might have been, it was sufficient for this project, given the priority to produce software to test the methodology rather than a full product.

### 4.4.3 Overview of the Interface

As can be seen from Figure 4.1 on page 35, the user interface for the Reactive Planner consists of two basic grid structures with a sliding separator down the middle.

The left-hand grid is used for data input – this is where milestone data is entered, including due dates and so on. The right-hand grid displays a Gantt-style chart, which is updated as the user adds or edits the milestone information.

It was important to allow the user to see the overall plan as they were adding milestones or updating durations, which is why the direct input approach was chosen for the main screen. This was a lesson learnt from a previous student's project [Richards, 2004], where the input had been implemented as a series of web forms, separate from the plan itself. Immediate response (i.e. the updating of the Gantt chart as soon as the user entered more milestone data) is also important, because during planning it is useful to be able to examine "what if?" scenarios [Kliem and Ludin, 1993]. Other aspects of the user interface all aim to be standard and therefore consistent.

The data input itself is very simple, following repeated feedback during the direct research stage that software packages such as MS Project offered too many options. Many of those interviewed reported that the interface often led them to setting exact deadlines or introducing arbitrary nonexistent dependencies in order to make the plans display as desired. Additionally, because the time dimension in MS Project (and similar) plans is very refined from the outset – usually set to days – the plans appear very precise from the beginning. This can often lead to a false sense of confidence in the estimates included in the plan, which later leads to exacerbated problems when these estimates turn out to be flawed.

For exactly this reason, the Dynamic Planner defaults to a coarse time dimension. Since the intention is that the user will first construct a low precision plan, consisting purely of milestones, the default (as shown) is that months are displayed. Equally, for large projects, the time dimension might be quarters or years. As the project progresses (and so the planning), the plan will graduate to a medium- and eventually high-precision version, adding sub-milestones and miniature milestones. As previously stated, miniature milestones should never represent less than a week's work, so the most refined timescale will correspond to weeks.

Nevertheless, it may still be necessary to mark a particular day – for instance, there is a clear deadline for this project and it is important to note the actual day and time, rather than just the relevant week. Flexibility is thus desirable in the representation of the plan itself. For this reason, there is an unstructured notes field, to be used as the user sees fit. In our given example, this would therefore be used to add in the details of when the dissertation should be handed in. Equally, the milestone description could also include the due date.

Colour coding of the milestone bars (which has not been implemented, but should be if the planner was to be developed into a proper user software package) is another feature designed to allow the user flexibility. One way in which it could be used would be resource allocation, achieved by assigned a particular colour to a person or resource. Another would be tracking of which milestones were completed by their original due date, perhaps by colouring those that overran in red.

Figure 4.1: Screenshot of the Reactive Planner (running on Linux)

## 4.5 System Implementation

The system was implemented in a series of iterations, each iteration containing a number of features matching the user stories in the requirements document. Any framework required for a particular functionality was implemented along with the relevant feature, in accordance with the Extreme Programming principle of building code only when it is actually needed [Wells, 2004, "No Functionality is Added Early"].

An example of this was the storage of the plan data – in the initial stages the planner simply used the GUI grid classes (wxGrid) to store data as well as handle the user input. Only when a separate internal representation of the plan data was needed (i.e. when the plan needed to be saved) was the *planData* class written to encapsulate all the plan data. This in turn is extendable to contain other details about milestones (e.g. colour that the milestone bars should be, hierarchy level) but this code will not be written until its associated functionality is developed.

There are two classes that make up the Dynamic Planner. The main class is the *PlannerGUI.py* class, that encapsulates the Controller and View aspects of the architecture. The application itself is an instance of this class and the constructor contains all the necessary code to create and combine the GUI widgets. A number of event handlers are then bound to the actual interface, so that when a particular menu option is pressed or other event occurs, the appropriate response is called. These event response functions are also augmented by a number of "helper functions" that perform minor tasks to aid the event response.

The secondary class is *planData.py* which contains all of the code to handle the internal representation of the data. Python has surprisingly flexible native data storage and so this is primarily a wrapper for a list of dictionaries – each dictionary represents a milestone and the Python list structure is akin to the arrays found in other languages[2]. The class handles data access, as well as storing to file (by "pickling" the data structure) and then reconverting saved files into the object representation needed.

Another impressive aspect of Python as a programming language is how much can be done with relatively little code. The planner software in total only consists of about 500 lines of code, which is remarkably little given the functionality. Admittedly, some of this is because the wxPython GUI libraries encapsulate a great deal of what is used. Nevertheless, the language itself was found to be very elegant and understandable – few code level comments are necessary, although of course classes and methods have been documented appropriately. In fact, Python has the *pydoc* utility which examines code for doc comments and auto-constructs documentation. All the code for this project has been written in accordance with the Python documentation guidelines and so *pydoc* produces very nice documentation for the classes.

[2] In fact, the Python list structure is quite advanced, automatically increasing the size to accommodate any number of items, etc. It is therefore more realistically comparable to an ArrayList structure from Java.

## 4.6 Testing

The approach taken during this project was heavily influenced by the Test-Driven Development approach advocated by Beck [2003]. Before any code was written, tests were written to specify when the functionality would work as intended. In the case of the GUI, these were largely user-oriented tests, in the form of scripts that a human tester would follow. For the data classes of the Dynamic Planner, unit tests were written.

Python does have a unit testing framework, in the shape of the *unittest* module (discussed by Pilgrim [2004, p.272-277]). However, since it was only appropriate to construct automated tests for the data management aspect of the planner, it was not necessary to use such a fully-featured tool. Instead, a Python "trick" was used : included in the *planData.py* class file is a section of code that will only run if the file is executed on its own (e.g. by running *python planData.py*) rather than when it is used by another class (i.e. when it is imported for use by *PlannerGUI.py*). This code specifies a number of unit tests for the data management code and displays results at the end to standard output, unless directed elsewhere.

The rest of the planner code was tested interactively, using scripts that specify how the functionality should behave. By their very nature, these scripts cover both unit and integration testing, since for the interface to behave in the manner expected Model, View and Controller sections of the code must be working correctly together. Examples of these scripts can be found in the appendices, along with an indication of whether the tests passed at the time of completion. Since best practice is to write tests before implementation has even begun, a number of test scripts are included for functionality that has not yet been implemented.

An obvious omission from this testing strategy is usability/interaction testing. Although this was acceptable in the development of a piece of research software, if the software was to be released to a group of users as a project planner then this failing would need to be remedied. The first step would be an expert evaluation to identify potential pitfalls in the interface as it stands. It might also be useful to examine the cognitive model that users form when using the software and compare it to that evidently desired by the methodology specification, to ensure a match. Ongoing development should include user involvement in the testing process, as well as an appropriate route for feedback on the design itself.

# Chapter 5

# Evaluation

The stated aim of the project was to not only develop a new planning methodology, suitable for flexible planning in our uncertain world, but also to construct software to be used to evaluate the new methodology. Thus, a discussion of the evaluation carried out and its results is an essential component of this dissertation. In this chapter we will discuss the evaluation of the Dynamic Planning methodology and separately that of the Dynamic Planner, as well as the technology used and process followed in developing it.

## 5.1 Evaluating the Dynamic Planning Methodology

The first stage of evaluation for the methodology consisted of a series of interviews. These were carried out with some of those who had participated in the direct research, as well as number of other professional project managers. First the background to the project was discussed, then the Dynamic Planning methodology was presented and finally a number of questions were asked about how useful and viable a technique might be in practice.

All felt that the methodology seemed to have merit and might be particularly useful for software projects. Some were very excited about the concept and eager to see whether the evaluation using the planning software would bear fruit. Others expressed concern that the milestone-based plans would not be able to completely replace the traditional task-based Gantt charts. This led to increased focus in the methodology description on the role of personal todo lists, action plans and augmented calendars.

The second stage of the methodology evaluation was to test how easy the methodology is to follow, as well as whether the plans produced are in fact more flexible than traditional alternatives. It was for this purpose that the Dynamic Planner was developed, so that test subjects could be asked to plan using software specifically geared to support the new methodology.

A number of subjects were recruited to participate in this second stage of the evaluation process. They had varying levels of experience managing projects, with the only prerequisite being that they understood the concepts of project planning and

had previously constructed plans for projects. One criticism that could be made of the group that did participate in this stage would be that very few had significant project management experience. Recommendations related to this fact appear in the Further Work section. On the other hand, there was an advantage in using subjects with less experience, as they proved less "set in their ways" and receptive to the new planning concepts presented.

Test subjects were given an informal general description of a project to be undertaken over a seven month period and told that they could assume that sufficient resources would be available to complete the project in this time. They were then asked to construct a plan for the project, in the traditional style that they would normally use. This was done using MS Project and then they answered a short questionnaire about this initial plan. They were then asked to read a one-page description of the Dynamic Planning Methodology and to then construct another plan for the same project, using the Dynamic Planner. After this they were again asked to fill out a short questionnaire, similar to the first.

The final stage of the test involved the subject being informed that a very significant change had occurred that would affect the project. They were then asked to adjust their plans appropriately, after which they again answered a number of questions. After the experiment had been completed, they were briefly interviewed and their thoughts on the methodology, software and the test itself were captured.

The results were very encouraging. All subjects found replanning significantly easier when using the Dynamic Planning method and software. They also all felt that the dynamic plans were thus more likely to be updated and so more useful for managing the project. Although the traditional Gantt charts are seen as easy to follow, all subjects indicated they felt that the dynamic plans would be even more so. A more detailed summary can be found in the appendices.

One mixed result was the ease of use of the Dynamic Planning methodology itself. Although some of the subjects found the new process natural and accessible, others found it more difficult. Some attributed the problems to an over-familiarity with the traditional planning process, others to the circumstances of the test, noting that because they had to first plan in a traditional manner and then immediately try to get into the alternative frame of mind needed for the new process, the Dynamic Planning stage might seem artificially difficult.

Whatever the cause, this result indicates that care needs to be taken when disseminating and training the new approach. Users need to be given time and guidance to acclimatise to the new way of thinking. Additionally, the Dynamic Planner itself will need to have a very comprehensive help facility, to provide both contextual help on the software functionality and the process itself. It might also be interesting to run some evaluations with completely inexperienced subjects who have not yet been taught or used the traditional planning method. This possibility is discussed in the Further Work section.

## 5.2 Evaluation of the Dynamic Planner

The Dynamic Planner successfully achieved its primary aim – to provide users with the ability to follow the Dynamic Planning methodology and construct a plan accordingly. Although some features were not implemented that would be required were the Planner intended to be used as a complete planning package, what has been implemented proved sufficient to evaluate the methodology, as seen in the previous section.

The processes followed in constructing the Dynamic Planner were successful as well. The combination of the Extreme Programming and Test-Driven Development approaches meant that a working version of the application was available at every stage, however limited its functionality might have been. The approach to testing was especially valuable, since full testing after every user story implementation meant that new bugs that were accidentally introduced were identified close to their creation, making debugging infinitely easier.

Although in hindsight the selection of Python and the wxPython GUI toolkit was probably still correct, there were some unanticipated problems when using the technologies. Python itself is a simple and elegant language and was relatively easy to learn. It is more often used for simple scripts than entire applications, however, so the learning curve was steep and the resources available to learn from sparser than if a more mainstream application development language (such as Java) had been selected.

wxPython, on the other hand, is quite complex and suffers from its origins – because it is essentially just a wrapper to the C++ wxWidgets library, wxPython programming is often quite different from best practice Python coding. Thus, in constructing the Dynamic Planner, it was necessary both to understand the Python idiom and then to adjust to a more traditional approach when including wxPython code.

Another issue encountered when working with wxPython was related to its status. wxPython is an open-source project, staffed by volunteers. Luckily it is under very active development, so there is an enthusiastic developer community and a high-traffic mailing list where questions can be asked. Unfortunately the flip side is that the documentation is often patchy and the active development means that there are often changes to the API (Application Programming Interface).

Development of the Dynamic Planner began using the 2.4.2 release of wxPython, but it soon became apparent that a number of the toolkit bugs being encountered would be solved by upgrading the toolkit. From then onwards, the 2.5.3 release was used, but during the development time of this project, there has been a new release of the toolkit on average every month. Since each release has subtle differences in the API and different bugs, it was felt best to stay with the 2.5.3 release, since the initial upgrade had resulted in a great deal of code needing to be refactored.

How ever much the upgrade helped with some problems, developing with wxPython was generally quite difficult. Out-of-date documentation meant that a number of questions had to be referred to the mailing list; the number of releases available then meant that multiple solutions needed to be explored before one that actually worked was finally found.

At the time of writing, however, a new stable release series (2.6) has just become available, which includes an API freeze. Hopefully this new release will make development with wxPython significantly easier, especially as there is also a project underway to create a new documentation resource. Ideally the planner should be converted to run on this before any additional development is carried out.

# Chapter 6

# Conclusion

This project has been a success, with all the objectives being met. We have successfully applied various advances in artificial intelligence planning to project planning, combining direct research with deduction from background reading to produce a new approach: Dynamic Planning. This appears to be the first time that artificial intelligence planning approaches have been applied to project management, although the opposite has happened in the past.

Additionally, software to support this planning methodology has been developed and used in the evaluation of the efficacy of the approach. Initial results indicate that Dynamic Planning is certainly more flexible than traditional project planning and potentially also contends extremely strongly in terms of the ability to track project progress. A solid foundation has been provided to develop the Dynamic Planner into a piece of end-user software, as well as to test the Dynamic Planning methodology further.

From a personal development point of view, I have learnt a great deal from undertaking this project. The research and academic writing skills gained will definitely be useful in future. Learning Python, wxPython and developing a desktop application for the first time were all challenging tasks, but enjoyable at the same time. The knowledge and understanding of AI and project planning has awakened a great interest in these and associated topics, which I hope to pursue in future. I hope to continue this work myself, as detailed in the Further Work section which follows.

## 6.1 Further Work

In this section we will discuss the next steps if the work is to be continued. Firstly we discuss potential next steps for the Dynamic Planning methodology itself and then the opportunities that exist for the planning software.

### 6.1.1 Dynamic Planning Methodology

The first area that deserves further attention is that of research and background reading. Since the primary focus of this project was the reapplication of AI planning

advances to project management, these two areas directed the background reading and research that was undertaken. However, during the course of the project it has become apparent that understanding the abilities of people is crucial in designing an appropriate methodology for flexible project planning.

Although care has been taken to base assumptions about people's behaviour and attitude to planning on the direct research, it would also be worthwhile to conduct some additional background reading in the areas of psychology and cognitive science. The intention here would be to validate underlying assumptions of the Dynamic Planning methodology, as well as identifying viewpoints that could be brought to bear on the flexible planning problem. In particular, investigations into the mental model of planning and the emotional and thought processes that occur during planning could prove fruitful.

As was noted when the methodology was originally set out, the focus has been on the project managers and the members of the project teams. Although their needs have been well-represented and catered for, there are of course others who make use of project plans. The needs of stakeholders who are not directly involved in the project teams, but nevertheless use the plans (for instance project sponsors, external vendors or customers, etc), must be investigated and the plan representation potentially adjusted to make the plans as useful to them as to the project teams.

Additionally, the assumption was made during this project that the Gantt chart was the optimal plan representation to be adapted for our use. This was based both on its ubiquity in the standard project management texts, but also on the direct research carried out during the course of the project. What this research in fact showed was that the Gantt chart is the most *popular* plan representation. This made it an obvious choice for our use in a new methodology, since when introducing new concepts it is often useful to tie them to very familiar ones. Nevertheless, additional research to investigate whether the Gantt chart is in fact the *best* plan representation for flexible planning still needs to be carried out.

The initial evaluation of the Dynamic Planning methodology was both useful and the results encouraging, but much remains to be done. Due to the obvious limitations of evaluating in an experimental setting (only so much can be done in an hour, after all!), the potential of this approach has only been touched upon. Additional testing and evaluation is definitely needed and the following should be investigated:

- *Testing with different groups of users.* The evaluation already carried out was done by a mixed group of relatively inexperienced project planners. It would be interesting to carry out similar evaluations with completely inexperienced students, to see if they understand and adopt the methodology more readily, not having already learnt and used the traditional approach. Testing with experienced project managers would also be desirable, because they may be better placed to evaluate the true usefulness of the approach, based on their past experience.

- *Testing in the real world.* Since the Dynamic Planning methodology is designed to help project managers deal with the uncertain nature of the world, the next logical step is to test it in this environment. However useful testing the

> approach experimentally may be from an academic viewpoint, the true test of a planning approach will be when it is being used for the entirety of a real project. Making this possible should be the eventual aim of any further work.

Carrying out this kind of testing of course requires the Dynamic Planner as support. Additional work to be carried out in developing this software must therefore be discussed next.

### 6.1.2 Dynamic Planner

Since the requirements elicitation extended beyond the basic functionality needed to perform the initial evaluation of the Dynamic Planning methodology, the roadmap for development has already been drawn. Although to perform the testing with different groups of users described above, little additional development would be necessary, in order to release the methodology into the real world and test its use in real projects, significantly more work needs to be completed on the Dynamic Planner.

Firstly, a number more features need to be implemented. The most crucial of these in terms of the methodology support are of course the provision of various levels of milestone (i.e. sub-milestones, miniature milestones, etc, in a hierarchy), the ability to switch time dimensions (allowing for a steady progression from a low-precision towards higher-precision versions of a plan) and the provision of colour-coding for the milestone bars. There are of course other features that are not essential to the methodology, but useful or required in a business setting nonetheless.

An obvious example is the ability to export plans to alternative formats, so that they can be viewed by users who do not have the Dynamic Planner installed. The software has been implemented with this in mind – since the plan data is stored separately in native Python data structures, pulling this data out into an alternative format (for instance, HTML or PDF or CVS) should be relatively easy. Various other features are also detailed in the full Requirements document found in the appendices.

During the implementation of the fuller feature set, it would be prudent to engage in usability evaluations, to ensure that the interface being implemented is intuitive and in line with Human Computer Interaction best practice. An expert evaluation of the interface would be a useful first step, to identify any major problems, and thereafter a user-centred design approach should be employed.

From a technical point of view, the Dynamic Planner needs to be adapted to use the newly released 2.6 series of the wxPython GUI toolkit. Since the API for this release has been frozen, ongoing development should be much easier and more predictable. Additionally, the current code was written for Python 2.3, but since 2.4 is the latest stable version, it may also be worth updating the software to this newer version and taking advantage of any new functionality or speed enhancements.

Finally, in order to distribute the Dynamic Planner more widely, more easily installable versions are required. Currently to run the software, the user must already have installed both Python (2.3) and wxPython (2.5.3) separately. Although thanks to the work of both development communities, installing Python and wxPython is

remarkably easy, it is still not wise to expect users to manage software dependencies on their own.

For this reason, creating a Windows .exe file, Mac app folder and Linux package for the Dynamic Planner would be useful. There are existing tools that make this process easy. *py2exe* is a tool for creating Windows binaries that contain all Python modules needed for the application in one executable. *py2app* and *bundlebuilder* are similar utilities for creating standalone binaries for Macintosh. For Linux the *freeze* utility included with Python may suffice, but in order to promote wider use it would probably be wise to supply Debian .deb archives, Red Hat RPMs and similar packages for the other popular Linux distributions.

# Bibliography


Philip E. Agre and David Chapman. What are Plans For? *A.I. Memo 1050a, MIT Artificial Intelligence Library*, October 1989. URL `http://hdl.handle.net/1721.1/6487`.

Phillip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272, 1987.

Erling Anderson, Kristoffer Grude, Tor Haug, and J Rodney Turner. *Goal Directed Project Management*. Kogan Page, 1988.

Erling S Anderson. Warning: Activity Planning is Hazardous to Your Project's Health! *International Journal of Project Management*, 14(2):89–94, 1996.

Kent Beck. *Test-Driven Development by example*. Addison-Wesley, Boston, 2003.

Kent Beck. *Extreme Programming Explained : Embracing Change*. Addison Wesley, 2 edition, 2004.

Kent Beck and Martin Fowler. *Planning Extreme Programming*. The XP Series. Addison-Wesley, Boston, 2004.

Thomas R. Block and J. Davidson Frame. *The Project Office*. Crisp Learning, 1998.

Barry W. Boehm. A Spiral Model of Software Development and Enhancement. *Computer*, 21(5):61–72, May 1988.

R. A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

Mark Brown. *Successful Project Management*. Hodder & Stoughton, 1998.

Joanna J. Bryson and Lynn Andrea Stein. Modularity and Design in Reactive Intelligence. *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 1115–1120, August 2001.

David Buchanan and Andrzej Huczynski. *Organizational Behaviour: An introductory text*. Financial Times Prentice Hall, Harlow, 2003.

Tristan Caulfield and Joanna Bryson. Simulating Chunking in Computer Chess using Clustering.

Prodromos D Chatzoglou and Linda A Macaulay. A review of existing models for project planning and estimation and the need for a new approach. *International Journal of Project Management*, 14(3):173–183, 1996.

Alistair Cockburn. *Agile Software Development*. Pearson Education, Boston, February 2002a.

Alistair Cockburn. *Writing Effective Use Cases*. Addison-Wesley, Boston, 2002b.

Tom DeMarco. *The Deadline: A Novel About Project Management*. Dorset House Publishing, 1997.

Tom DeMarco. *Slack: Getting past burnout, busywork, and the myth of total efficiency*. Broadway Books, 2002.

Jesse James Garrett. AJAX: a new approach to web applications. *http://www.adaptivepath.com/publications/essays/archives/000385.php*, 18 February 2005.

John C. Goodpasture. *Quantitative Methods in Project Management*. J Ross Publishing, 2004.

Steve Grand. *Creation: life and how to make it*. Harvard University Press, Cambridge MA, 2001.

Java Website. http://java.sun.com.

Jython Website. http://www.jython.org/.

Ralph L Kliem and Erwin S Ludin. *The Noah Project*. Gower Publishing, 1993.

Ralph L Kliem and Irwin S Ludin. *The People Side of Project Management*. Gower Publishing, 1992.

G.E. Krasner and S.T. Pope. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. *Journal of Object Oriented Programming*, 1(3):26–49, 1988.

Dennis Lock. *The Essentials of Project Management*. Gower Publishing Company, 2001.

Steve McConnell. *Rapid Development: Taming Wild Software Schedules*. Microsoft Press, 1996.

Drew McDermott. A Heuristic Estimator for Means-End Analysis in Planning. *In Proceedings, AIPS '96*, 1996.

Douglas McGregor. *The Human Side of Enterprise*. McGraw-Hill, New York, 1960.

Mark Pilgrim. *Dive Into Python*. Apress, New York, 2004.

Jennifer Preece, Yvonne Rogers, and Helen Sharp. *Interaction Design :: beyond human computer interaction*. John Wiley & Sons, Inc; New York, 2002.

Louise Pryor and G. Collins. Planning for contingencies: a decision-based approach. *The Journal of Artificial Intelligence Research*, 4:287–339, 1996.

Arie Quik. The RASCI Technique/Analysis. *http://www.qed.nl/rasci_ en.htm*, 11 January 2005.

Stephen Richards. A Reactive Project Planner. Dissertation submitted as part of BSc Maths and Computing degree at the University of Bath, 13 May 2004.

Stuart Russell and Peter Norvig. *Artificial Intelligence – A Modern Approach*. Prentice Hall, New Jersey, 2003.

M.J. Schoppers. Universal Plans for Reactive Robots in Unpredictable Environments. *IJCAI-87*, pages 1039–1046, 1987.

Ian Sommerville. *Software Engineering*. Pearson Education, 2001.

Joel Spolsky. Painless Software Schedules. *http://www.joelonsoftware.com/articles/fog0000000245.html* 9 January 2005.

Systems-Gap-Working-Party. *Closing the Gaps in Project Management Systems*. Butterworths, 1984.

The Standish Group. The Standish Group Report: Chaos. *http://www.projectsmart.co.uk/docs/chaos-report.pdf*, 5 January 2005.

Frank Tsui. *Managing Software Projects*. Jones & Bartlett Publishers, Canada, 2004.

Daniel S. Weld. An Introduction to Least Commitment Planning. *AI Magazine*, 15 (4):27–61, 1994.

Don Wells. Extreme Programming. *http://www.extremeprogramming.org/*, 18 October 2004. URL `http://www.extremeprogramming.org/`.

Meri Williams. Interviews and Conversations with Procter & Gamble Project Managers.

Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, Chichester, 2004.

wxPython Website. http://www.wxpython.org/.

wxWidgets Website. http://www.wxwindows.org/.

S Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3): 73–83, 1996.

# Part I

# APPENDICES

# Appendix A

# Review of Existing Systems

As existing planning methodologies were examined, it was important to be conscious of the existing software available to support these processes. Once the Dynamic Planning approach had been developed, it was important to identify whether existing software could support planning in this way. For this reason existing software was reviewed and the synopsis is included here for completeness.

## A.1 MS Project

MS Project is like a Swiss army knife of project planning. It can produce virtually any form of project plan, from Gantt charts (the default view) through to resource levelling views, task lists to network diagrams. In many ways this represents overkill for the average project. The interface, although reasonably user-friendly, does not invite the user to plan *with* the system. In practice, many users report that they plan on paper or in other simpler tools (such as Excel) and then input their plans into MS Project to display the appropriate format. The high degree of help and user guidance that MS Project provides also suggests that it is not intuitive enough to be used as a planning tool in the conceptual stage of a project, just as a plan representation tool later in the process.

MS Project does provide a number of useful scheduling aids – for instance, it will automatically define start and end dates if given duration and dependency information – but these are only useful when producing high precision plans. Indeed, the amount of detail needed about constraints in fact can hamper the planning process, since in order to produce even an outline plan arbitrary task lengths and deadlines must be input. This can lead to very precise (but incorrect) plans being drawn up at the project outset and can only contribute to the early commitment issues described by DeMarco [2002].

## A.2 Open Source Alternatives

There are a number of open source project management tools available for download. For the most part these tend to mimic the functionality provided by MS Project,

usually in three main areas: task lists, Gantt charts and resource utilisation & levelling diagrams. The following appear to provide similar (if more limited) functionality to MS Project in these areas:

- Mr Project (`http://mrproject.codefactory.se/`)
- JXProject (`http://www.jxproject.com/`)
- QtGantt (`http://www.gumbley.me.uk/qtgantt.html`)
- Open Workbench (`http://www.openworkbench.org/`)
- Gantt Project (`http://ganttproject.sourceforge.net/`)

Although arguably these (and various other similar programs) are doing well to replicate the most useful functionality from MS Project, there is a definite gap with regards to software for alternative planning approaches. At the moment UML or diagram drawing tools (such as Dia) could be used to produce Milestone Plans but no current project planning software appears to do so currently.

Since all the existing tools mimic MS Project and share its focus on task-based precision activity planning, none can be used in a manner that supports Dynamic Planning. For this reason it was essential to develop a new application (the Dynamic Planner), specifically to support the new planning methodology.

# Appendix B

# Requirements Document

This requirements documents consists primarily of User Stories, brief descriptions of functionality as encountered by a user of the system. This method of requirements representation was felt to be appropriate for two key reasons. Firstly, since there was no real customer for the system, a detailed requirements document was not needed to act as a basis of understanding between developer and customer. Secondly, the software was developed as a vehicle for testing the Flexible Project Planning methodology and so the priority was to support the planning processes of this methodology.

The document enclosed here covers all requirements set out during the course of the project. Some of these were not implemented, for reasons discussed in the main dissertation.

## B.1 High-Level Requirements

1. Provide support for plan production using the Results-Focused Planning methodology
   (a) Allow plans to be constructed from milestones
   (b) Allow plans of various levels of precision, as appropriate to the project stage
   (c) Display the plan graphically, in a similar style to a Gantt chart
   (d) Facilitate easy editing and updating of the plan
   (e) Facilitate progress tracking and timely issue identification
2. Software must be cross-platform, able to run on any of the three major platforms (Windows, Mac, *nix), so that eventually it can be distributed and tested by a wider audience
3. Software should be user-friendly and intuitive, requiring minimal training to construct plans
4. It must be possible to share plans between users and stakeholders

## B.2 User Stories

### Basic Functionality

1. User opens the application and is greeted with a blank plan
2. User adds a milestone by entering a milestone description and due date [the user may also indicate a duration – if not, duration is assumed to be 1 unit of time]
3. User edits an existing milestone by selecting it and editing one of its attributes directly
4. User deletes an existing milestone by selecting Delete Row from the Edit menu
5. User inserts a new row (and milestone) by selecting Insert Row from the Edit menu
6. User views the plan [which is in fact continuously updated and visible, as the milestone data is entered and edited by the user]
7. User adds a note to a milestone
8. User saves the plan to disk
9. User opens a plan from disk
10. User exits the application

### Intermediate Functionality

1. User prints the plan
2. User exports the plan to an alternative format [HTML, XML or PDF, for instance]
3. User changes the hierarchy level of a milestone [i.e. making it a sub-milestone, or a sub-sub-milestone, or the reverse]
4. User requests and receives general help on the methodology and the software

### Advanced Functionality

1. User changes time dimension [to move plan from being low-precision (e.g. expressed against quarters) to a higher precision plan (e.g. expressed against months or weeks) – the system should auto-convert the milestone data appropriately]
2. User selects a particular colour for a milestone [a shortlist of frequently used colours should be available on the toolbar for easy access]
3. User requests and receives context-sensitive help on either the methodology or the software

# Appendix C

# Test Scripts

Although Unit Testing was an appropriate vehicle for testing the data storage aspect of the application (i.e. the Model), all the other functionality required user interaction and so an alternative testing method was required. This consisted primarily of scripted user testing. The scripts used during this testing are included here for completeness.

| User Story | Test Description | Status |
|---|---|---|
| Add A Milestone | Enter milestone description :: ensure that no change is made to Gantt chart | Pass |
| | Enter milestone description, due date :: ensure that Gantt chart is updated with milestone bar against due date | Pass |
| | Enter milestone description, due date, duration :: ensure that Gantt displays milestone bar that spans the number of time periods indicated by the duration with the right-hand edge against the due date | Pass |
| Edit A Milestone | Edit milestone description :: ensure that no change is made to Gantt chart | Pass |
| | Edit due date :: ensure that Gantt chart is updated with milestone bar against newly inputted due date | Pass |
| | Edit duration :: ensure that Gantt chart is updated with milestone bar that spans the number of time periods indicated by the duration, with the right-hand edge against the due date | Pass |
| Insert a Row | Select Insert Row from the Edit menu :: ensure that row is added where the cursor lies, to both the data input grid and the Gantt chart grid | Pass |

| User Story | Test Description | Status |
|---|---|---|
| | Select a row in the data input grid and then select Insert Row from the Edit menu :: ensure that row is added at point where row was selected and not where cursor lies | Pass |
| Delete a Row | Select Delete Row from the Edit menu :: ensure that the row where the cursor lay is deleted | Pass |
| | Select a row in the data input grid and then select Insert Row from the Edit Menu :: ensure that the row which was selected is deleted (and not the row where the cursor lay) | Pass |
| View Plan | Ensure that the data entry grid and contained information is visible | Pass |
| | Ensure that the Gantt chart accurately reflects the information in the data input grid and is visible | Pass |
| Open From File | Select Open from the File menu :: ensure that Open File... dialog appears | Pass |
| | Select a file in the Open File... dialog and click OK :: ensure that the plan data is loaded successfully | Pass |
| | With a file already open, select Open from File menu, choose another file :: ensure that the current information is cleared and replaced with the newly loaded data | Pass |
| Save To File | Create a new plan and select Save from the File menu :: ensure that the Save As... dialog appears | Pass |
| | Open an existing plan, edit it, then select Save from the File menu :: ensure that file is actually saved and that save confirmation dialog appears | Pass |
| Add a Note to a Milestone | Add a note in the Note field of the milestone :: ensure it is displayed | Pass |
| Exit the Application | Select Quit from the File menu :: ensure that the application closes | Pass |

| User Story | Test Description | Status |
|---|---|---|
| Shortcuts Work | For each shortcut, perform the shortcut key sequence :: ensure that the same functionality is evident as if the menu option had been chosen directly | Pass |
| Print the Plan | Select Print from the File menu :: ensure that the plan is sent to the printer | Fail |
| Export the Plan | Select Export from the File menu :: ensure that file dialog is displayed, allowing user to specify the file format | Fail |
| | Once file has been exported :: check that exported data matches current plan | Fail |
| Increase Hierarchy Level of a Milestone | Whilst cursor or selection is on an existing milestone, select Increase Indent from Edit menu :: ensure that milestone description is preceded by appropriate dashes to show it is a sub-milestone | Fail |
| | Whilst cursor or selection is on a blank row, select Increase Indent from Edit menu :: ensure that milestone description is preceded by appropriate dashes to show it is a sub-milestone | Fail |
| | Whilst cursor is in milestone description, type an even number of dashes to indicate hierarchy level :: ensure that the saved data reflects the appropriate milestone hierarchy level | Fail |
| Decrease Hierarchy Level of a Milestone | Whilst cursor or selection is on an existing milestone, select Decrease Indent from Edit menu :: ensure that milestone description has 2 dashes removed from the start | Fail |
| Request Context Sensitive Help | Perform a task and select Help With This from the Help menu :: ensure that context-sensitive help is displayed | Fail |
| Request General Help | Select Help from the Help menu :: ensure that general help information is shown | Fail |

| User Story | Test Description | Status |
|---|---|---|
| Change Time Dimension | With an existing plan, select Change Time Dimension from the Edit menu :: ensure that Change Time Dimension dialog is displayed | Fail |
| | In the Change Time Dimension dialog, select a new time dimension :: ensure that application appearance, plan data and plan graph are all converted to the new dimension | Fail |
| | Decrease precision of time dimension :: ensure that the more precise data is still stored and that the application converts only for display | Fail |
| Colour a Milestone | Press button in the Colour column alongside an existing milestone :: ensure that Colour Chooser Dialog is displayed | Fail |
| | In Colour Chooser Dialog, select a colour :: ensure that plan is updated to show milestone in that colour and that colour information is stored internally | Fail |
| | Ensure that in Colour Chooser Dialog, frequently used colours are easily selectable | Fail |
| | Ensure that in the Colour Chooser Dialog, user can define their own colours | Fail |
| Run Application on Any Platform | Run the application on Windows :: ensure that all tests pass | Pass |
| | Run the application on Linux :: ensure that all tests pass | Pass |
| | Run the application on Macintosh :: ensure that all tests pass | Fail |
| Distribute the Application to Any Platform | Ensure that an easy method of installing and running the application is available for Windows, Linux and Macintosh | Fail |

As mentioned in the main body of this dissertation, the data classes of the application were tested using built in unit tests. A screenshot of the test results can be found in Figure C.1 on the following page.

Figure C.1: Plan Data Test Results

# Appendix D

# Dynamic Planner Screenshots

In the following pages, a number of screenshots of the Dynamic Planner are presented, running under both Linux and Windows. Unfortunately a Macintosh computer was not available to test the software under.

Figure D.1: Main Screen, running on Linux

Figure D.2: About Dialog, running on Linux

Figure D.3: Save As Dialog, running on Linux

Figure D.4: Save Confirmation, running on Linux

Figure D.5: Main Screen, running on Windows

Figure D.6: File Menu (example of menus), running on Windows

Figure D.7: Open Dialog, running on Windows

# Appendix E

# Evaluation Results

| Question | Key Results |
|---|---|
| How easy do you think the Gantt chart you constructed will be to follow? | 4/6 say easy to follow |
| How useful do you think the Gantt chart you constructed will be for tracking progress during the project? | 5/6 say not very useful |
| Do you think the Gantt chart you constructed would be followed? | All say unlikely to be followed, with various strength in the statement |
| How easy was the Dynamic Plan to construct? | 3 say easy to construct, 3 express that it was somewhat difficult |
| If you did not find it easy to construct, why do you think this is? | All agree the issue is in the change of paradigm from task-based to result-based thinking |

| Question | Key Results |
|---|---|
| Do you think this plan will be more or less easy to follow than the traditional Gantt chart? | All feel easier to follow |
| Do you think this plan will be more or less useful for tracking progress than the traditional Gantt chart? | All feel dynamic plan much more useful for progress tracking |
| Which plan was easier to update? (given the significant change) | 6/6 found the dynamic plan easier to update |
| If this situation had occurred in your work environment, do you think the traditional plan would have been updated? | Only one respondent felt they would have updated the original plan |
| Do you think using the Dynamic Planning method would make you more likely to keep your plans up-to-date? | 6/6 say yes |

# Appendix F

# Code Listings

```
from pickle import *

"""
Part of the framework for the Dynamic Planner planning software
Produced by Meri Williams for CM30076 (Final Year Project)

Author:    Meri Williams
Version:   1.1
"""
class planData:
    """Holds the plan data, storing both the text input by the user into the right-hand grid
       and the other information stored by the system about the displayed plan (e.g.  colour,
       submilestone level, etc).  The data stored here should mirror additions and changes
in
       the milestone information and the plan itself.

       The data stored is what is needed to construct the graphical plan, not the graph
       itself

       When run as a standalone .py file (python planData.py), a series of unit tests are
       executed and the results displayed"""

    def __init__(self):
        """Constructor initializes array to hold the rows (dictionary structures) of the plan
        data"""
        self.data = []

    def getNumRows(self):
        """Returns the number of rows stored in the planData instance"""
        return len(self.data)

    def addRow(self, rowData):
        """Adds the given row of data to the end of the list"""
        self.data.append(rowData)

    def insertRow(self, rowNo, rowData):
        """Inserts the given row of data at the index specified"""
        if rowNo < len(self.data):
            self.data.insert(rowNo, rowData)
        else:
            print "Sorry, cannot add value at that index"

    def editRow(self, rowNo, rowData):
        """Edits the given row, replacing existing data with that which is supplied, else
        creating a new row at the end"""
        if rowNo < len(self.data):
            self.data[rowNo] = rowData
        else:
            self.addRow(rowData)

    def fetchRow(self, rowNo):
        """If the row exists (i.e.  rowNo < number of rows), returns the value stored at
        rowNo index.  Otherwise returns False"""
        if rowNo < len(self.data):
```

```
54              return self.data[rowNo]
55          else:
56              print "Sorry, that row does not exist"
57              return False
58
59      def clearRow(self, rowNo):
60          """Clears the value of the given row, setting it to an empty string "" """
61          if rowNo < len(self.data):
62              self.data[rowNo] = ""
63          else:
64              print "Sorry, that row does not exist and so cannot be cleared"
65
66      def deleteRow(self, rowNo):
67          """Deletes the row from the list entirely (removing the object stored there)
68
69          Returns the object that was stored there, or False if the row did not exist"""
70          if rowNo < len(self.data):
71              return self.data.pop(rowNo)
72          else:
73              return False
74
75      def saveToFile(self, filename):
76          """Saves the planData object to a file called filename, as a serialized object
77          using Pickle.
78
79          Returns true if successful, false if any exceptions are raised"""
80          try:
81              savefile = open(filename, 'w')
82              saver = Pickler(savefile)
83              saver.dump(self)
84              savefile.close()
85              return True
86          except IOError, PickleError:
87              print "Failed to save to file"
88              return False
89
90      def loadFromFile(self, filename):
91          """Loads a file from filename, returning the planData object"""
92          try:
93              loadfile = open(filename, 'r')
94              loader = Unpickler(loadfile)
95              plan = loader.load()
96              loadfile.close()
97              return plan
98          except IOError, PickleError:
99              print "Failed to open file"
100              return False
101
102      def printAllRows(self):
103          """Prints out all the data stored in each row in a human-readable form.
104          Primarily for debugging purposes"""
105          for i in range(0, len(self.data)):
106              print "Line", i, "::", self.fetchRow(i)
107
```

```
108 if __name__== "__main__":
109     """Series of unit tests to ensure that functionality is working as required.
110 
111     These will be executed when run as a standalone file (i.e.  >> python planData.py)"""
112     # Initial result values to ensure no false positives
113     rowAdded = False
114     rowEdited = False
115     nonexistentRowEdited = False
116     rowCleared = False
117     rowDeleted = False
118     fileSaved = False
119     fileOpened = False
120     PROBLEM = False
121     rowsCounted = False
122     testsPassed = 0
123 
124     testplan = planData()
125     #test adding a row
126     testplan.addRow(10)
127     if testplan.fetchRow(0) == 10:
128         rowAdded = True
129         testsPassed += 1
130     #test editing a row
131     testplan.editRow(0, 11)
132     if testplan.fetchRow(0) == 11:
133         rowEdited = True
134         testsPassed += 1
135     testplan.editRow(3, 20)
136     if testplan.fetchRow(1) == 20:
137         nonexistentRowEdited = True
138         testsPassed += 1
139     #test deleting a row
140     testplan.clearRow(0)
141     if testplan.fetchRow(0) == "":
142         rowCleared = True
143         testsPassed += 1
144     compare = testplan.fetchRow(1)
145     testplan.deleteRow(0)
146     if testplan.fetchRow(0) == compare:  #comparing to value of what used to be at index 1
147         rowDeleted = True
148         testsPassed += 1
149     #test exporting to savefile
150     testplan.addRow({'name':'meri', 'age':22})
151     testplan.addRow({'name':'elly', 'age':23})
152     if testplan.saveToFile('savetest.txt'):
153         fileSaved = True
154         testsPassed += 1
155 
156     #test importing from savefile
157     loadtest = planData()
158     extratest = planData()
159     extratest.addRow(46664)
160     loadtest = testplan.loadFromFile('savetest.txt')
161     if loadtest.fetchRow(1) == testplan.fetchRow(1):
```

```
        fileOpened = True
        testsPassed += 1
    if loadtest.fetchRow(0) == extratest.fetchRow(0):
        PROBLEM = True
        testsPassed -= 1
    else:
        testsPassed += 1
    if extratest.getNumRows() == 1:
        rowsCounted = True
        testsPassed += 1

    print "TEST RESULTS:"
    print "Row added:  ", rowAdded
    print "Row edited:  ", rowEdited
    print "Nonexistent row edited:  ", nonexistentRowEdited
    print "Row cleared:  ", rowCleared
    print "Row deleted:  ", rowDeleted
    print "File saved:  ", fileSaved
    print "File opened again:  ", fileOpened
    print "Rows counted:  ", rowsCounted
    print "Do we have a problem?  ", PROBLEM
    print "Number of tests passed:  ", testsPassed

    print "TESTPLAN:"
    testplan.printAllRows()
    print "LOADTEST:"
    loadtest.printAllRows()
    print "EXTRATEST:"
    extratest.printAllRows()
```

```
1 import wxversion
2 wxversion.select('2.5.3')     # Selects the version of wxPython to use (multi-versions poss)
3 import os
4 from wxPython.wx import *
5 from wxPython.grid import *
6 from planData import *
7
8 """
9 Dynamic Planner GUI classes
10 Contains View & Controller aspects of the application
11 Produced by Meri Williams for CM30076 (Final Year Project)
12
13 Author:    Meri Williams
14 Version:   2.1
15 """
16
17 # wxNewId() gives unique identifier to each of the constants, for this run of the program
18 ID_ABOUT=wxNewId()
19 ID_SAVE=wxNewId()
20 ID_SAVE_AS=wxNewId()
21 ID_OPEN=wxNewId()
22 ID_INSERT_ROW=wxNewId()
23 ID_DELETE_ROW=wxNewId()
24 ID_INCREASE_INDENT=wxNewId()
25 ID_DECREASE_INDENT=wxNewId()
26 ID_NEW=wxNewId()
27 ID_CLOSE=wxNewId()
28 ID_EXIT=wxNewId()
29 ID_HELP=wxNewId()
30 ID_PRINT=wxNewId()
31 ID_EXPORT_PDF=wxNewId()
32
33 # Constants to make it easier to adapt the GUI
34 MILESTONE_LABEL = "Milestone"
35 ENDTIME_LABEL = "Due"
36 ESTIMATE_LABEL = "Duration"
37 NOTE_LABEL = "Notes"
38
39 # Some hard-coded values to assist in a simple time plan
40 INITIAL_ROWS = 100
41 INITIAL_COLS = 12
42 year = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
43
44 # MAIN GUI CODE ==========================================================================
45 class PlannerGUI(wxFrame):
46     """The primary class that provides the Graphical User Interface for the planning
47     application, encompassing both the View and Controller aspects of the architecture.
48
49     The __init__function prepares the interface, creating the menus, key widgets and so
50     on (i.e.  the View) and binding events to associated functions
51
52     A variety of functions containt the actual functionality of the interface, forming
53     the Controller aspect of the architecture.  Uses the planData class to handle the
54     application data (i.e.  the Model)"""
```

```
55      def __init__(self,parent,id,title):
56          """Creates the interface, building up the widgets to form the GUI and associates
57          events with event handling functions"""
58          # first call the parent constructor
59          wxFrame.__init__(self,parent,wxID_ANY, title, size=wxDefaultSize ,style=wxDEFAULT_FRAME_STYLE|wxN
60          self.CreateStatusBar() # A StatusBar in the bottom of the window
61
62          # MENU CODE ----------------------------------------------------
63          # Create the menus
64          filemenu = wxMenu()
65          editmenu = wxMenu()
66          helpmenu = wxMenu()
67          # Add the menu items, shortcuts and their status bar info strings
68          # NOTE: \t denotes the keyboard shortcut
69          filemenu.Append(ID_NEW, "&New\tCtrl+N"," Create a new plan")
70          filemenu.Append(ID_OPEN, "&Open\tCtrl+O", " Open an existing plan")
71          filemenu.Append(ID_SAVE, "&Save\tCtrl+S", " Save the current plan")
72          filemenu.Append(ID_SAVE_AS, "Save &As...", " Save the current plan")
73          filemenu.Append(ID_CLOSE, "&Close", " Close the current plan")
74          filemenu.AppendSeparator()
75          filemenu.Append(ID_PRINT, "&Print", " Print the current plan")
76          filemenu.Append(ID_EXPORT_PDF, "&Export to PDF", " Export the current plan to PDF")
77          filemenu.AppendSeparator()
78          filemenu.Append(ID_EXIT,"E&xit\tCtrl+Q"," Terminate the program")
79
80          editmenu.Append(ID_INSERT_ROW, "&Insert a Row", " Insert a row")
81          editmenu.Append(ID_DELETE_ROW, "&Delete Row", " Remove the selected row")
82          editmenu.Append(ID_INCREASE_INDENT, "&Increase Indent", " Increase indentation to allow
submilestones")
83          editmenu.Append(ID_DECREASE_INDENT, "D&ecrease Indent", " Decreases indentation")
84
85          helpmenu.Append(ID_ABOUT, "&About"," Information about this program")
86          helpmenu.Append(ID_HELP, "&Help", " Help using this program")
87
88          # Create the menu bar and then add the menus to it
89          menuBar = wxMenuBar()
90          menuBar.Append(filemenu,"&File")
91          menuBar.Append(editmenu, "&Edit")
92          menuBar.Append(helpmenu, "&Help")
93
94          self.SetMenuBar(menuBar)   # Adding the MenuBar to the overall Frame
95          self.splitter = wxSplitterWindow(self, -1) # Creates a splitter window (div in 2)
96
97          # GRID STUFF ----------------------------------------------------
98          # Create the input grid and do some formatting
99          self.grid = wxGrid(self.splitter,-1)
100          self.grid.CreateGrid(INITIAL_ROWS,4)
101          self.grid.SetColLabelValue(0, MILESTONE_LABEL)
102          self.grid.SetColLabelValue(1, ENDTIME_LABEL)
103          self.grid.SetColLabelValue(2, ESTIMATE_LABEL)
104          self.grid.SetColLabelValue(3, NOTE_LABEL)
105
106          # Create specific attributes so that only relevant values can be entered in certain columns
107          endTimeAttr = wxGridCellAttr()
```

```
108         endTimeAttr.SetEditor(wxGridCellChoiceEditor(year)) # makes the editor for cells a drop-
down list
109         self.grid.SetColAttr(1, endTimeAttr)
110
111         estimateAttr = wxGridCellAttr()
112         estimateAttr.SetEditor(wxGridCellNumberEditor())
113         self.grid.SetColAttr(2, estimateAttr)
114
115         self.grid.SetMargins(0,0)
116         #self.grid.SetRowLabelSize(0)
117         #self.grid.AutoSize() # fits the grid to the column & row labels
118         self.grid.SetColSize(0,250)
119         self.grid.SetColSize(1,100)
120
121         self.plan = wxGrid(self.splitter, -1)
122         self.plan.CreateGrid(INITIAL_ROWS, INITIAL_COLS)
123         for i in range(0,INITIAL_COLS):
124             self.plan.SetColLabelValue(i, year[i])
125
126         self.plan.SetMargins(0,0)
127         self.plan.SetRowLabelSize(0)
128         self.plan.SetDefaultColSize(50)
129         self.plan.EnableEditing(False)
130         self.plan.EnableGridLines(True)
131         self.splitter.SetMinimumPaneSize(100)
132         self.splitter.SplitVertically(self.grid, self.plan, 600)
133         self.splitter.Fit()
134
135         # SIZER CODE ----------------------------------------------------
136         self.sizer = wxBoxSizer(wxHORIZONTAL)
137         self.sizer.Add(self.splitter,1,wxEXPAND)
138         self.SetSizer(self.sizer)
139         self.SetAutoLayout(1)
140         self.sizer.Fit(self)
141
142         # EVENTS CODE ---------------------------------------------------
143         # Binds events to the functions that should be called when they occur
144         EVT_MENU(self, ID_OPEN, self.OnOpen)
145         EVT_MENU(self, ID_SAVE, self.OnSave)
146         EVT_MENU(self, ID_SAVE_AS, self.OnSaveAs)
147         EVT_MENU(self, ID_EXIT, self.OnExit)
148         EVT_MENU(self, ID_INSERT_ROW, self.OnInsertRow)
149         EVT_MENU(self, ID_DELETE_ROW, self.OnDeleteRow)
150         EVT_MENU(self, ID_ABOUT, self.OnAbout)
151         EVT_GRID_CELL_CHANGE(self, self.OnCellChange)
152         EVT_KEY_DOWN(self.grid, self.OnKeyDown)
153
154         self.Show(true)
155         print(self.splitter.GetBestSize()) #DEBUG
156         """ NOTE: The GetBestSize function is returning inappropriate results at the moment
157         This is a bug in wxPython itself.  The workaround used is that the application
158         immediately maximises itself when loaded.  An upcoming release of wxPython may
159         include a fix and so ideally the software should eventually be upgraded"""
160         self.Maximize()
```

```
161
162         # Initialise some of the key variables that will be used by the program
163         self.plandata = planData()
164         self.savepath = ""
165
166     # EVENT HANDLER RESPONSES ==========================================================
167     def OnSave(self,evt):
168         """ When the user selects Save (either using Ctrl-S shortcut or the menu, then
169         the application first identifies whether there is an existing savepath (i.e.
170         whether there is already a file open).
171
172         If this is the case, then the updated plan is saved to the file already open and
173         a confirmation is shown that the file has been saved successfully.
174
175         If the plan has not already been saved, the Save As response is called."""
176         if self.savepath != "":
177             self.plandata.saveToFile(self.savepath)
178             #print "Saved to file"
179             saveConfirm = wxMessageDialog( self, " File successfully saved!","File Saved!",
wxOK)
180             saveConfirm.ShowModal()
181             saveConfirm.Destroy()
182         else:
183             self.OnSaveAs(evt)
184
185     def OnSaveAs(self, evt):
186         """Creates a file chooser dialog for the user to choose where they would like to
187         save their file.  The user can choose to view either just dynamic plans or all
188         files.  Once a filename has been chosen, the plan data is saved."""
189         wildcard = "Dynamic plan (*.plan)|*.plan|"     \
190                    "All files (*.*)|*.*"
191         savedialog = wxFileDialog(self, message="Save file as...", defaultDir=os.getcwd(),
192                       defaultFile="", wildcard=wildcard, style=wxSAVE)
193         if savedialog.ShowModal() == wxID_OK:
194             self.savepath = savedialog.GetPath()
195             print('You selected "%s"'% self.savepath)
196             diditsave = self.plandata.saveToFile(self.savepath)
197             print "Just tried to save to file"
198             print "Successfully saved?  ", diditsave
199             self.plandata.printAllRows()
200         savedialog.Destroy()
201
202     def OnOpen(self, evt):
203         """Creates a file dialog for the user to choose the file they wish to open.  If
204         the user selects a file, then it is loaded and the plan data is displayed"""
205         wildcard = "Dynamic plan (*.plan)|*.plan|"     \
206             "All files (*.*)|*.*"
207         openfiledialog = wxFileDialog(self, "Please choose file to open", os.getcwd(), "",
wildcard, wxOPEN)
208         if openfiledialog.ShowModal() == wxID_OK:
209             openpath = openfiledialog.GetPath()
210             #clear the internal storage and wipe the grids clean before loading
211             self.plandata = planData()
212             for row in range (0, self.grid.GetNumberRows()):
```

```
                self.clearRow(self.grid, row)
                self.clearRow(self.plan, row)
            self.plandata = self.plandata.loadFromFile(str(openpath)) #load the plan data
            self.savepath = openpath #update the savepath to the file just opened
            self.loadPlanData()
        openfiledialog.Destroy()

    def OnAbout(self,evt):
        """ When the About menu option is selected, this function displays a
        small dialog to give some basic information about the software"""
        aboutMessage = wxMessageDialog( self, " Dynamic Planner \n"
                            " \"Flexible Planning in a Changing World\" \n \n"
                            " Provides basic support for the \n"
                            " Dynamic Planning Methodology developed \n"
                            " by Meri Williams & Joanna Bryson","About", wxOK)
        aboutMessage.ShowModal()
        aboutMessage.Destroy()

    def OnExit(self,evt):
        """ Closes the application when Exit is selected from the File Menu"""
        self.Close(True)
        """NOTE: This should be updated to eventually check if the user has saved what
        they have been working on and if not to prompt them to do so"""

    def OnInsertRow(self,evt):
        """Inserts a row into the data and plan grids.  Where the row is inserted depends
        on where the cursor is or what row/cells the user has selected.  Selection
        takes precedence over simple cursor position"""
        blankrow = {'milestone':"", 'due':"", 'estimate':"", 'owner':""}
        if self.grid.IsSelection():
            selectedrows = self.grid.GetSelectedRows()
            row = selectedrows[0]
        else:
            row = self.grid.GetGridCursorRow()
        self.grid.InsertRows(pos=row, numRows=1, updateLabels=True)
        self.plandata.insertRow(row, blankrow)
        self.plan.InsertRows(pos=row, numRows=1, updateLabels=True)

    def OnDeleteRow(self, evt):
        """Deletes a row from the data and plan grids.  Which row is deleted depends
        on where the cursor is or what row/cells the user has selected.  Selection
        takes precedence over simple cursor position"""
        if self.grid.IsSelection():
            selectedrows = self.grid.GetSelectedRows()
            row = selectedrows[0]
        else:
            row = self.grid.GetGridCursorRow()
        self.grid.DeleteRows(pos=row, numRows=1, updateLabels=True)
        self.plandata.deleteRow(row)
        self.plan.DeleteRows(pos=row, numRows=1, updateLabels=True)

    def OnCellChange(self,evt):
        """Is called when a cell is edited by the user.

```

```
        Updates the graphical plan based on the data the user has input into the data grid
        on the left hand side"""
        row = evt.GetRow()
        self.storeRow(row)
        self.plandata.printAllRows()
        self.repaintRow(row)

    def OnKeyDown(self, evt):
        """Remaps the enter key, so that if editing a row, the cursor moves right
        when enter is pressed and if at the end of the row, jumps to the beginning
        of the next row.

        Adapted from the demo code supplied with wxPython"""
        if evt.KeyCode() != wx.WXK_RETURN:
            if evt.KeyCode() == wx.WXK_DELETE:
                # Placeholder: should remap Del to clear row if row is selected (TODO)
                evt.Skip()
                return
            evt.Skip()
            return
        if evt.ControlDown():    # the edit control needs this key
            evt.Skip()
            return
        self.grid.DisableCellEditControl()
        success = self.grid.MoveCursorRight(evt.ShiftDown())
        if not success:
            newRow = self.grid.GetGridCursorRow() + 1
            if newRow < self.grid.GetTable().GetNumberRows():
                self.grid.SetGridCursor(newRow, 0)
                self.grid.MakeCellVisible(newRow, 0)
            else:
                # this would be a good place to add a new row if your app
                # needs to do that
                pass

    # HELPER METHODS ============================================================
    def clearRow(self, whichgrid, row):
        """Clears the plan row of any existing graph data, so new can be painted"""
        for col in range(0, whichgrid.GetNumberCols()):
            whichgrid.SetCellBackgroundColour(row, col, wxWHITE)
            whichgrid.SetCellValue(row, col, "")

    def repaintRow(self, row):
        """Repaints the plan row, to reflect changes the user has made

        This is the key Controller method for updating the displayed plan based on
        changed to the Model"""
        self.clearRow(self.plan, row)
        estimate = self.grid.GetCellValue(row, self.findColumnIndex(self.grid, ESTIMATE_LABEL))
        endTime = self.grid.GetCellValue(row, self.findColumnIndex(self.grid, ENDTIME_LABEL))
        endTimeCol = self.findColumnIndex(self.plan, endTime)
        # Tries to get estimate value, but if this fails defaults to 1
        try:
            estimate = int(estimate)
```

```
321         except ValueError:
322             print("Couldn't convert to int, setting estimate to 1")
323             estimate = 1
324         if isinstance(estimate, int):
325             if (estimate > 0) and (endTimeCol != False):
326                 for i in range(estimate):
327                     self.plan.SetCellBackgroundColour(row, endTimeCol-i, wxGREEN)
328         self.plan.ForceRefresh()
329 
330     def findColumnIndex(self, whichgrid, label):
331         """Finds the number of the column with the supplied label"""
332         numCols = whichgrid.GetNumberCols()
333         for i in range (0, numCols):
334             if label == whichgrid.GetColLabelValue(i):
335                 return i
336         return False
337 
338     def storeRow(self, row):
339         """Stores the row information as a dictionary in the planData object"""
340         rowContents = {}
341         #NOTE: TODO update so these call the findcolindex function instead of hardcoded
342         rowContents["milestone"] = str(self.grid.GetCellValue(row, 0))
343         rowContents["due"] = str(self.grid.GetCellValue(row, 1))
344         rowContents["estimate"] = str(self.grid.GetCellValue(row, 2))
345         rowContents["owner"] = str(self.grid.GetCellValue(row, 3))
346         self.plandata.editRow(row, rowContents)
347 
348     def loadPlanData(self):
349         """For use when plandata has just been updated directly, usually when the user
350         has loaded a file and so the plan needs to be displayed"""
351         # NOTE TODO update so these call the findcolindex function instead of hardcoded
352         for i in range(0, self.plandata.getNumRows()):
353             rowdata = self.plandata.fetchRow(i)
354             self.grid.SetCellValue(i, 0, rowdata["milestone"])
355             self.grid.SetCellValue(i, 1, rowdata["due"])
356             self.grid.SetCellValue(i, 2, rowdata["estimate"])
357             self.grid.SetCellValue(i, 3, rowdata["owner"])
358             self.repaintRow(i)
359 
360 # MAIN METHOD LOOP ================================================================
361 if __name__== "__main__":
362     planner = wxPySimpleApp()
363     frame = PlannerGUI(None, -1, "Dynamic Planner")
364     planner.MainLoop()
```